

LABORATOIRE L3I
Pôle Sciences et Technologie
17042 La Rochelle Cedex 1 - FRANCE

ANTHONY HERBE
Université de la Rochelle
Sciences et Technologie
Année 2005

RAPPORT DE STAGE DE 2IÈME ANNÉE DE MASTER PROFESSIONNEL IMA

Mise en place d'une plate-forme d'indexation d'images par le contenu

M.A.D.O.N.N.E
MAsse de DOnnees issues de la Numerisation du patrimoiNE

Enseignant tuteur : Jean-Marc OGIER
Maître de stage : Jean-Marc OGIER

UNIVERSITÉ DE LA ROCHELLE

Master Informatique, Mathématiques
et leurs Applications
Mention Génie Informatique



LABORATOIRE L3I
Pôle Sciences et Technologie
17042 La Rochelle Cedex 1 - FRANCE

ANTHONY HERBE
Université de la Rochelle
Sciences et Technologie
Année 2005

RAPPORT DE STAGE DE 2IÈME ANNÉE DE MASTER PROFESSIONNEL IMA

Mise en place d'une plate-forme d'indexation d'images par le contenu

Maître de stage : Jean-Marc OGIER

Enseignant tuteur : Jean-Marc OGIER



LABORATOIRE L3I
Pôle Sciences et Technologie
17042 La Rochelle Cedex 1 - FRANCE

ANTHONY HERBE
Université de la Rochelle
Sciences et Technologie
Année 2005

RÉSUMÉ :

Ce rapport a été réalisé lors de mon stage de fin d'études de Master à l'université de la Rochelle. Celui-ci s'est déroulé au sein du laboratoire de la Rochelle (L3i) du 7 février au 7 juillet 2005.
Le but de ce stage concernait la mise en place d'une plate-forme web d'indexation et de recherche de documents picturaux dans le cadre du consortium M.A.D.O.N.N.E.

Ce rapport est donc composé de trois parties :
– Le contexte du stage ;
– Les problématiques de développement de la plateforme ;
– La réalisation de la plateforme.

MOTS-CLEFS :

Plateforme web, reconnaissance des formes, indexation d'images, signatures, inter-opérabilité, XML

Remerciements

Je tiens tout d'abord à remercier mon maître de stage et tuteur, Mr Jean-Marc OGIER, pour m'avoir proposé un sujet aussi intéressant et formateur et de m'avoir guidé dans mon travail tout en me laissant un maximum de libertés et d'initiatives. Je tiens encore à le remercier pour toutes les démarches administratives concernant l'obtention de mes outils de travail (prêt de portable, etc...).

Je tiens également à remercier Aymeric BLONDEL pour m'avoir épaulé dans la mise en place de l'environnement de développement, de test et de production ; Philippe HARRAND pour tous ses conseils concernant mes différents soucis sous Unix, Surapong UTTAMA, Mouhamed HAMMOUD, Manuel BARRAUD et tous ceux que j'aurais pu oublier pour leurs conseils et suggestions durant la durée de mon stage.

Enfin, je remercie ma compagne qui m'a soutenu (et supporté) durant toute cette période ainsi que ma famille pour m'avoir permis de faire cette dernière année d'études.

Table des matières





Liste des illustrations



Liste des tableaux

Liste des codes sources

Introduction

Afin de valider l'année de Master de la formation IMA de l'université de la Rochelle, chaque étudiant doit effectuer un stage de vingt deux semaines.
Ce stage est une étape importante pour un étudiant, non seulement du point de vue de la scolarité, mais aussi d'un point de vue personnel.
Ce rapport présente l'ensemble des travaux que j'ai effectué au cours de mon stage au laboratoire L3i (Laboratoire Informatique, Image, Interaction) de la Rochelle en Charente-Maritime.

Durant ces vingt deux semaines, l'activité du stagiaire doit se partager entre une activité d'apprentissage "fondamental" (on pourrait dire "théorique") et une activité d'apprentissage "professionnel".

L'objectif de cette expérience est de parvenir à une réflexion sur des problématiques qu'engendre une plate-forme d'indexation et de recherche d'informations de type pictural en se basant sur une expérience concrète qu'est le développement d'une solution logicielle.

Cette réflexion a pour but de faire ressortir les approches et solutions possibles pour la résolution de ces problématiques.

Ce rapport s'articulera donc autour de trois axes :

- Tout d'abord, nous présenterons le cadre du stage, le laboratoire L3i, le sujet proposé et les problématiques posées.
- Ensuite, nous examinerons ces problématiques et tenterons de leur apporter des solutions.
- Enfin, le reste du rapport portera sur l'ensemble du travail effectué : la méthodologie, les choix effectués, les résultats obtenus, leur analyse et les perspectives envisageables.

Ce rapport, produit avec LaTeX 2ε, a été compilé le 1er juillet 2005.

Première partie

Contexte du stage

Chapitre 1

Présentation du L3i.

1.1 Son histoire

Le laboratoire L3i est reconnu par le Ministère de la Recherche comme équipe d'accueil (EA2118) depuis 1997. Il regroupe désormais une quarantaine d'enseignant-chercheurs (dont 12 habilités à diriger des recherches), essentiellement issus de la communauté informatique (section 27 du CNU) et génie informatique (section 61) de l'Université de La Rochelle. Au premier janvier 2003, 18 doctorants y sont inscrits.

Le laboratoire a changé de nom : d'« Informatique et Imagerie Industrielle », il s'est tourné vers « Informatique, Image, Interaction ». Il se positionne donc beaucoup plus sur une image, facteur d'interactions que sur les aspects technologiques associés à une informatique industrielle. Cette orientation a été motivée par une volonté d'aborder, de la manière la plus cohérente, complète et fondamentale, les différentes facettes de chaque contexte applicatif auquel le laboratoire est confronté. En particulier, l'ouverture du laboratoire aux autres domaines disciplinaires (les arts, le littoral, les usages) a montré que la prise en compte des interactions, sous toutes les formes qu'elles peuvent prendre, était primordiale.

L'Image et le Comportement sont donc au centre de l'activité du laboratoire. Une partie de son activité concerne donc naturellement les aspects analyse/modélisation/-synthèse - c'est l'objet de la thématique « Image et séquences d'Images » (ISI). Pour l'analyse d'images et pour l'interprétation de comportements complexes, un travail de fusion/classification est nécessaire. La thématique « Modélisation, Analyse, Traitement, Recherche d'Informations Complexes et Évolutives » (MATRICE) traite ce problème et le replace dans un cadre plus général, lui permettant d'aborder, en particulier, les données semi-structurées. La thématique « Génie Logiciel- Architecture, Méthodes Modèles, Outils Langages, Évaluation » (GéLoAMéMOLÉ) contribue à inscrire les travaux menés dans un contexte opérationnel et un cadre méthodologique. Ceci permet de mieux comprendre, à travers leur modélisation, les systèmes représentés par des images, mais aussi, de produire les logiciels associés à nos résultats en les intégrant dans une architecture pertinente.

Les quatre dernières années ont permis :

– de provoquer des synergies entre l'IUT et l'UFR de Sciences afin d'unifier les chercheurs en informatique de l'université de La Rochelle. Ceci a permis de restructurer les recherches autour de thématiques dans lesquelles tous les chercheurs ont leur place. Il en résulte que le L3i est désormais le plus gros labora-



toire de l'Université de La Rochelle.
– de développer des contacts industriels permettant de motiver des recherches fondamentales et appliquées.
– de s'insérer dans les réseaux de recherche, participer aux réseaux industriels de nos domaines, de préparer l'insertion dans les réseaux européens.
– d'améliorer la diffusion scientifique, en qualité et en quantité.
– de trouver une "liberté financière" en multipliant par quatre les recettes du laboratoire, tout en limitant les concessions sur la nature des recherches.
– de mener une politique de recrutement coordonnée et de limiter les déstabilisations consécutives à des recrutements relativement nombreux, d'accroître le nombre de doctorants (6 par an depuis les 2 dernières années).
– de mettre en place une politique de valorisation avec dépôts de brevets et transfert de technologie.

1.2 Sa structuration

Au niveau de la dynamique du laboratoire, afin de ne pas souffrir, comme beaucoup d' entre eux, des barrières induites par des structurations administratives et scientifiques, une réflexion a été menée sur une structuration adaptée à sa taille et correspondant à la dynamique interne de celui-ci. L'activité de la recherche peut ainsi être décrite à l'aide d'une matrice permettant de croiser les **thématiques** scientifiques et les **projets** :

– Une **thématique** correspond à une vision académique d'une problématique scientifique. Elle concerne une communauté de recherche identifiée. Un chercheur appartiendra donc, de manière privilégiée à une thématique. C'est le vecteur dominant. Cependant, les activités de recherches pluri-thématiques constituent un moyen d'organiser les recherches situées aux interfaces entre les thématiques. Elles sont justifiées dès lors qu'un chercheur s'investit dans un domaine disciplinaire et aborde les aspects méthodologiques ou s'il développe une théorie fondamentale, qu'il spécialise dans un contexte disciplinaire particulier.
– Un **projet** est le support de la transversalité entre thématiques scientifiques. Chaque **projet** ne se résume pas à un contexte applicatif. Le L3i a plutôt considéré des « objets », caractérisés par des contraintes particulières et une classe de problèmes associés, sur lesquels il était nécessaire de capitaliser des connaissances. Le laboratoire discrétise les recherches en fonction des différents problèmes à aborder. Chaque thématique est alors le cadre pertinent pour maîtriser l'ensemble des recherches menées dans le contexte des projets associés. Il est ainsi possible de replacer les travaux en utilisant le corpus des résultats disponibles, de développer de manière la plus générale possible de nouveaux résultats et enfin d'identifier les recherches spécifiques à mener.

Les cellules de la matrice ainsi formée contiennent des **Actions** qui correspondent à des recherches menées par des chercheurs, spécialement regroupés pour atteindre l'objectif visé. En fonction de cet objectif et de son envergure, une action peut impliquer plusieurs thématiques ou plusieurs projets. Une action est classiquement associée à un **contrat** avec un partenariat impliquant des industriels et/ou des académiciens. Le financement est alors principalement externe. Une action peut aussi consister en

une **recherche prospective**, définie en interne au laboratoire (après validation par le conseil scientifique). Le financement est alors pris en charge sur fonds propres du laboratoire. De telles actions correspondent à des recherches exploratoires menées au sein de celui-ci. Naturellement, des partenaires peuvent être associés. Les recherches prospectives se prolongent naturellement en contrats.

Il apparaît alors clairement que cette structuration induit une dynamique toute particulière puisque chaque action nouvelle est l'occasion, dans le cadre d'un séminaire interne puis, au sein du conseil scientifique, d'identifier les problématiques scientifiques sous-jacentes (**thématiques**) et les caractéristiques particulières à prendre en compte (**projets**). En outre cette organisation permet de provoquer les synergies nécessaires entre chercheurs pour atteindre les résultats visés. Les équipes sont ainsi brassées, de sorte que chacun puisse intervenir, dans son domaine de recherche, dans un contexte différent qui peut susciter de nouvelles recherches et l'enrichir du point de vue des autres chercheurs.

Il serait donc réducteur d'assimiler la notion d'**équipe** à celle de **thématique**, et tout aussi faux que d'adopter le point de vue des **projets**. Chaque chercheur se positionne naturellement dans la **matrice** en fonction de ses compétences. Thématiques et projets ne sont donc pas des entités administratives, mais scientifiques. Chacune est animée par un responsable qui gère uniquement l'animation et la cohérence des activités scientifiques menées au sein de son entité. La coordination générale et la gestion administrative sont assurées au niveau de la direction du laboratoire.

Ce mode de fonctionnement est possible, d'abord parce que le laboratoire est d'une taille idéale : pas trop important pour imposer une structuration trop lourde, suffisamment riche pour aborder des problèmes complexes à l'aide de compétences multiples. On permet aux chercheurs de se concentrer sur les tâches scientifiques, en les dégageant au mieux des charges administratives. Le dynamisme des chercheurs est aussi un atout prépondérant dans la réussite de ce mode de fonctionnement. Enfin, ce mode de fonctionnement, basé sur l'interaction entre chercheurs, facilite l'ouverture vers des partenaires extérieurs au laboratoire.

1.3 Ses thématiques scientifiques

Image et séquences d'Images (ISI)

- Détection et suivi d'objets sur des séquences d'images ;
- Étude et modélisation de déformations ;
- Restauration et extraction de l'information ;
- Reconstruction, production et suivi de trajectoires ;
- Interaction et planification de séquences de traitement d'images.

Problématique : La finalité de cette thématique s'inscrit dans la problématique du traitement de l'information, restreint aux composantes « Traitement du Signal, Traitement de l'Image, Reconnaissance des Formes, Modélisation, et Conception d'algorithmes ». La thématique est, par ce fait, en parfaite cohérence avec les deux autres thématiques du laboratoire. De manière conjointe avec celles-ci, les travaux ont pour finalité d'ap-



porter des solutions originales et robustes sur le plan algorithmique : l'information pertinente extraite est combinée dans l'intention

1. de fabriquer des paramètres sur différents niveaux sémantiques, permettant à des techniques de décision multi-composantes d'évaluer la sortie de la chaîne de traitement ;
2. de définir des invariants ou signatures (spatio-temporelles) servant d'indices pour l'indexation d'images ou de documents dans des systèmes d'informations ;
3. de développer une interaction dynamique entre les différents niveaux de la chaîne d'information pour optimiser la chaîne de traitement et d'analyse, cibler les traitements selon les classes d'objets, définir des stratégies dépendantes du contexte et inférer des attributs non accessibles (occultation, image dégradée).

La thématique "image et séquences d'images" s'est d'abord focalisée sur la problématique du traitement des images acquises à partir de systèmes d'imagerie de type industriel. Cette problématique s'est donc décliné très rapidement sur le mode restauration, extraction d'informations pertinentes et détection selon les spécificités industrielles : conditions d'acquisition difficiles, contraintes d'éclairement, contraintes de temps réel, image de haute définition. Cette thématique s'est par la suite enrichie de réflexions portant sur des domaines d'imageries multi-composantes (couleur : films anciens, images sous-marines, contexte routier, multi-canaux : imagerie satellitaire, multi-modalités : imagerie biomédicale, multi-formes : document (image, graphique, texte)) et s'appuyant sur des séquences temporelles (films anciens, séquences de trafics routiers, séquences de vues d'aquarium, séquences de mouvements dansés, imagerie biomédicale. . .) et/ou sur des contextes de systèmes d'informations documentaires (patrimoine, SIG,. . .). Le triplet restauration, extraction d'information et détection d'objets se situe résolument au cœur de l'activité thématique dans un contexte de caractérisation et d'interprétation du mouvement, des déformations et des relations spatio-temporelles des objets.

Modélisation, Analyse, Traitement, Recherche d'Informations Complexes et Évolutives (MATRICE)

- Classification robuste,
- Fusion de données à des niveaux d'abstraction et de structuration variables,
- Modélisation pour l'analyse de données spatio-temporelles,
- Modélisation pour une représentation multi-résolution des données spatiales géoréférencées

Problématiques : Un des objectifs de cette thématique scientifique est de contribuer au développement de systèmes de transformation et de structuration d'informations complexes évolutives. Les données d'entrée de ces systèmes peuvent être de niveaux de structuration variables (du faiblement structuré à la base de données), être numériques et/ou symboliques, et surtout concernent des processus évolutifs, pour lequel l'aspect temporel nécessite d'être modélisé (aspect dynamique de la mise à jour des données). Tous les éléments de la chaîne sont intégrés dans ces systèmes, depuis l'analyse des données, jusqu'à leur présentation à un utilisateur émetteur de requête.



Les problématiques scientifiques sous-jacentes concernent la classification, la modélisation pour l'analyse mais aussi pour la présentation des données, et l'extraction de caractéristiques, pour le développement de systèmes de recherche d'informations. Modéliser un système d'analyse et de recherche d'informations nécessite une indexation, c'est-à-dire une représentation du contenu de l'information ainsi que des connaissances du domaine couvert par le corpus, et une correspondance entre la requête et la représentation du document numérique analysé, correspondance permettant un calcul de la pertinence de la réponse. Sur un plan applicatif, les corpus d'analyse sur lesquels les recherches ont été menées concernent essentiellement des images, des séquences d'images ou des données environnementales.

S'appuyant sur des liens étroits avec les thématiques **ISI** et **GéloaMemoLé**, les recherches portent sur le développement d'outils d'analyse d'images et d'extraction de primitives, la recherche de méthodes robustes de classification pour l'indexation et l'interprétation, la fusion d'informations à divers niveaux d'abstraction (données, primitives, décisions) et de structuration (numérique/numérique, numérique/symbolique, symbolique/symbolique). Pour les données semi-structurées, on traite le cas particulier de la modélisation de données spatio-temporelles et de leur interaction avec les données environnementales, de données géo-référencées dans un contexte mobile.

Génie Logiciel - Architecture, Méthodes Modèles, Outils Langages, Évaluation (GéLoAMéMOLÉ)

– Modélisation de comportements d'agents et de leurs interactions dans un contexte de répartition et de mobilité.
– Vérification de leurs propriétés comportementales.
– Définition de modèles et de méthodes pour évaluer et contrôler la qualité de spécifications et du logiciel.
– Conception d'architectures logicielles à composants fiables et réutilisables.
– Modélisation de systèmes d'informations, d'informations complexes et de leur transfert.

Problématique : Les objectifs de la thématique sont la **conception de modèles, de méthodes, de langages et d'outils** pour représenter, analyser et mettre en œuvre le **comportement des systèmes complexes, mobiles et évolutifs**.

Le point de vue du L3i est de systématiquement développer une *approche générique*, dans chacun de ses secteurs d'activité informatique : *modélisation, méthodologie, prototypes*. C'est à dire que les modèles, les stratégies, les méthodes et les outils sont conçus dans le souci de permettre leur adaptation et leur *réutilisation* pour des cas d'étude, des domaines applicatifs et des contextes différents. Un autre point de vue systématiquement adopté est de chercher à *garantir la qualité* des méthodes, analyses adoptées ainsi que des produits logiciels fournis. Cela passe par un souci de *spécification et de documentation précises* des domaines, des besoins, des méthodes et des solutions logicielles. Cette spécification permet de *valider et vérifier par le test et le raisonnement formel* les applications mises en œuvre dans les prototypes. Elle répond également à la volonté mentionnée plus haut de travailler sur des solutions génériques réutilisables.



Les activités de recherche sont organisées en trois classes de préoccupations complémentaires :

- Modélisation et conception : Intégration de modèles, modèles de comportements, définition et validation de composants, automatisation de production de code ou de modèles intermédiaires, modélisation à base de systèmes multi-agents, modèles de données pour les systèmes de visualisation d'informations spatiales embarquées, modélisation des systèmes d'informations géo-référencées.
- Démarche de conception : Travaux sur la cohérence métrique des modèles produits et sur la méthodologie de production, critères métriques et stratégie qualité.
- Production d'outils théoriques et cadre méthodologique : treillis de Galois et ordres partiels, systèmes implicatifs.

Ces préoccupations définissent les activités de chercheurs du laboratoire. Chaque préoccupation se décline en différentes problématiques.

Chapitre 2

Sujet et objectifs

2.1 Présentation du contexte

2.1.1 Masse de données issues de la numérisation du patrimoine

« Le patrimoine culturel et scientifique de l'Europe est un bien public unique qui représente la mémoire collective et vivante de nos différentes sociétés et qui forme une base solide pour le développement des industries s'appuyant sur le contenu numérique dans une société de la connaissance durable. »

La communauté internationale (gouvernements, organismes...) ressent un besoin grandissant de sauvegarder ce patrimoine et de démocratiser l'accès à celui-ci. Les intérêts sont nombreux (nous pouvons citer quelques domaines : l'enseignement, l'industrie du tourisme, les médias...).

C'est ici qu'intervient le projet M.A.D.O.N.N.E [1] qui a pour mission de mettre en valeur différents biens du patrimoine international et plus particulièrement les ouvrages, les collections d'images et autres documents iconographiques. A court terme, ces nombreux documents constitueront une source gigantesque d'information (masse de données).

Considérant la grande quantité d'information à stocker, il est nécessaire de se pencher sur certains problèmes :

- La consultation en mode image des documents patrimoniaux suppose leur archivage et exige donc d'examiner de manière approfondie les possibilités spécifiques de compression de ces masses de documents.
- Se pose aussi le problème de simplifier au maximum la recherche d'un document. L'idée est de déterminer des indices s'adaptant aux différentes représentations de l'information que l'on peut rencontrer dans les documents patrimoniaux comme des zones textuelles, des images, des illustrations graphiques... Ces indices apportent des connaissances spécifiques qui aideront à la navigation.

[1] MAsse de DONnées issues de la Numérisation du patrimoiNE

Les différents laboratoires (L3i [2], LI [3], IRISA [4], PSI [5], CRIP5 [6], LORIA [7], LIRIS [8]) membres du projet M.A.D.O.N.N.E travaillent actuellement sur les problèmes liés à sa réalisation afin d'obtenir à terme une chaîne de traitements pour la valorisation d'une collection.

– Enfin, il est important de souligner que certains facteurs (culturels, sociaux ou économiques) risquent toutefois d'empêcher d'exploiter pleinement le potentiel de ces ressources, du moins dans un premier temps.

C'est pour ces raisons que ce projet vise aussi et en outre à renforcer les actions de sensibilisation sur les problèmes de conservation des données et par conséquent à générer des investissements et à établir une politique européenne commune sur l'utilisation du contenu culturel déjà numérisé.

2.2 Présentation du projet

Ce projet s'inscrit dans une démarche de sauvegarde et de valorisation de données patrimoniales dont la communauté internationale a pris conscience, comme en attestent les impulsions prises au niveau européen par les représentants nationaux des grands organismes de gestion du patrimoine. Le contexte de l'étude concerne les collections d'ouvrages numérisés, qui constitueront à très court terme d'énormes entrepôts de données, représentés sous forme d'images numérisées pour lesquelles les techniques traditionnelles des bases de données sont inopérantes. L'exploitation et la valorisation à venir de ces collections d'images n'ont toujours pas trouvé de réponses satisfaisantes, du fait même de leur caractère faiblement structuré. La génération de ces entrepôts de données, présentés sous forme de collections de documents hétérogènes faiblement structurés, soulève le problème de la recherche d'informations et de la navigation au sein de ces corpus.

Il s'agit de déterminer des indices s'adaptant aux différentes représentations de l'information qu'on peut rencontrer dans ces documents patrimoniaux comme des zones textuelles, imprimées ou manuscrites, des images, des illustrations graphiques. Ces signatures apporteront des connaissances spécifiques qui aideront la navigation et la recherche d'informations.

Précisément, dans le cadre de cette étude, il s'agira de mettre en place un démonstrateur des techniques de calculs de signatures sur des images graphiques, permettant d'indexer les images et de faciliter la recherche d'informations dans ces images.

L'idée est de mettre en place toutes les structures nécessaires en terme de Web, d'IHM, de représentation de données XML, de plate-forme de traitement, pour faire ce démonstrateur, en appui sur toutes les techniques issues du consortium MADONNE.

[2]Laboratoire de la Rochelle
[3]Laboratoire de Tours
[4]Laboratoire de Rennes
[5]Laboratoire de Rouen
[6]Laboratoire de Paris 5
[7]Laboratoire de Nancy
[8]Laboratoire de Lyon

Chapitre 3

Les problématiques d'indexation et de recherche d'informations sur des images dans une plate-forme Web

3.1 Recherche et traitements d'informations

Les informations d'une image relèvent de la perception de l'individu. En effet, à partir d'une image, l'individu peut s'intéresser à un ensemble de formes, de couleurs ou de composantes de l'image (textures, contours, etc...). Toutes ces informations constituent alors l'identité de l'image, ce qui la différencie ou la rapproche d'une autre image. Lorsque l'on cherche à retrouver des images proches d'une autre image, en réalité, l'individu effectue une recherche de similarités entre les composantes de ces images.

Prenons comme exemple le cas concret d'une image représentant une joueuse de beach-volley en pleine action. L'utilisateur peut vouloir rechercher des images qui contiennent un ballon, auquel cas, on s'intéressera à la forme caractéristique du ballon (un cercle) ou bien il peut vouloir rechercher toutes les scènes se déroulant sur une plage, auquel cas on fixera la recherche sur de grandes étendues de sable qui peuvent se caractériser par une certaine texture (granularité et couleur jaunâtre représentative du sable) ou enfin vouloir effectuer une recherche de toutes les photos représentant ce joueur de beach-volley, celui-ci se caractérisant par un visage bien défini (traits, couleur des yeux, texture de peau, organisation spatiale des yeux, nez, oreilles et bouche, etc...).



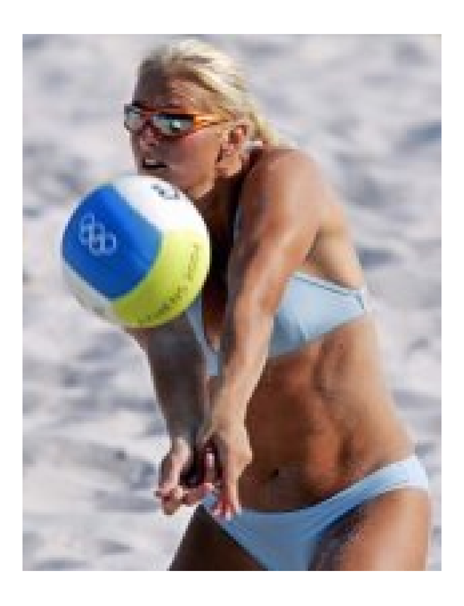


FIG. 3.1 – Exemple d'image



FIG. 3.2 – Extraction d'un élément d'une image

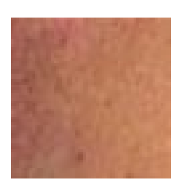


FIG. 3.3 – Exemple de texture d'un élément d'une image (texture de peau)

Il est donc important d'extraire toutes ces informations afin de pouvoir les comparer entre elles dans le but d'obtenir des résultats aux types de requêtes sus-citées. Cependant, de nouvelles problématiques apparaissent. Comment extraire, représenter et comparer ce type d'informations ?



3.1.1 La notion de signatures

Une signature caractérise un élément représentatif d'un sujet, il fournit une partie de son identité, qui le rend à la fois unique et reconnaissable. Malgré cette unicité, une signature peut être soumise à une comparaison avec une autre, afin de réaliser une recherche de similarités.

La première étape consiste à calculer ce type de signatures, ce qui implique bien évidemment de déterminer la façon de la représenter.
Le calcul de signatures dépend aussi des informations dont on dispose en entrée de celui-ci. En fait, une fois numérisée, une image n'est plus qu'un ensemble de pixels [1] entassés dans un unique fichier accompagnés le cas échéant de méta-données diverses qui varient selon le format du fichier choisi.

Outre les données contenues dans les méta-données du fichier, il semble logique que les signatures soient calculées à partir de l'information réelle, à savoir les pixels. Ainsi, les signatures seront donc le résultat d'un ensemble de calculs effectués sur les valeurs des pixels de l'images. Plus précisément, puisque un pixel est un ensemble de valeurs numériques, une signature sera donc lui aussi un ensemble de valeurs numériques.

Caractéristiques d'une « bonne » signature

Il existe de nombreux calculs permettant d'obtenir une signature sur une image. Cependant, il est nécessaire de faire le point sur ce que l'on doit attendre d'une signature intéressante.

Premièrement, une signature doit être suffisamment représentative d'une ou plusieurs informations de l'image. C'est-à-dire que celle-ci doit être suffisamment invariante pour que cette dernière ne soit pas valable uniquement dans un contexte bien précis et peu reproductible.

Reprenons l'exemple de l'image représentant la joueuse de beach-volley et supposons qu'on dispose d'une signature caractérisant l'organisation spatiale des zones caractéristiques du visage. Posons maintenant l'hypothèse qu'on possède une seconde image avec la même joueuse de beach-volley mais cette fois-ci dans une position différente (ex. : de face au lieu de trois quart). Si la signature est pertinente, la mesure de similarités entre les deux images en se basant sur cette signature devrait renvoyer une valeur très importante puisqu'il s'agit du visage de la même personne.

On peut donc en conclure que pour qu'une signature soit pertinente, celle-ci doit calculer des descripteurs qui soient invariants à la rotation. Il en va de même pour une transformation de l'image, basée sur le changement d'échelle ou la translation d'un « objet/sujet » dans l'image.

Enfin tant qu'il se peut, une « bonne » signature doit être un minimum sensible au

[1] Pixel : unité de base d'une image numérique. C'est le point minimal adressable par le contrôleur vidéo. Chaque pixel est en fait composé d'un ensemble de composants électrolumininescents



bruitage de l'image ou toutes interventions dues à une différence :
- de contraste,
- d'éclairage,
- de luminosité,
- d'espace couleurs,
- ....



FIG. 3.4 – Exemple de variations d'une même image

Malheureusement, il n'est pas toujours possible de trouver de telles signatures et certaines signatures qui paraissent intéressantes peuvent être sensibles à l'un ou plusieurs de ces critères.

3.1.2 Combinaison de signatures

Nous avons vu dans la section précédente qu'une image peut être décomposée en un ensemble d'éléments qui peuvent être identifiables et donc caractérisables. Il est alors possible de « signer » une image en calculant une ou plusieurs signatures sur chacun de ces éléments. La mesure de similarité entre images se fait alors par une distance entre la combinaison des deux ensembles de signatures.

Précédemment nous avons étudié les signatures sur des composants de l'image en tant qu' « objets » de l'image (ex. : ballon, volleyeuse, sable, etc...) mais il peut être intéressant d'utiliser des signatures sur des couches de l'image (ce qui implique évidemment un pré-traitement de segmentation de l'image) afin par exemple d'isoler les informations relatives aux contours, à la texture ou même des zones uniformes...



Cependant, il faut rappeler que l'acteur principal du système n'est autre qu'un être humain. Il en résulte alors que la nature de la recherche effectuée et la pertinence du résultat retourné par le système reste très subjectives. Il est donc très difficile d'adapter le moteur de recherche pour qu'il corresponde à l'intention de l'utilisateur. Pour s'adapter au mieux à ce type de situations, le système doit pouvoir apprendre et réagir face à cette subjectivité. Ainsi, l'utilisateur tient un rôle clé dans le système puisque, pour obtenir un résultat satisfaisant, il pourra influencer le système et plus précisément sur la pertinence des résultats.

Ce genre d'approches est donc basé sur le contrôle de pertinence (ou relevance feedback), technique qui permet de poursuivre une recherche d'information en la précisant à partir des résultats d'une requête antérieure.

Pour résumer, une image s'identifie à la fois dans sa globalité (l'ensemble des éléments qui la compose) mais également par chacun de ses éléments. On observe donc l'intérêt de pouvoir calculer une ou plusieurs signatures globales ainsi qu'un ensemble de signatures associées aux « éléments » (objets, couches, etc...) qui la composent. Néanmoins, quel que soit la combinaison des informations qu'on pourrait préciser dans la requête, le problème majeur de la subjectivité de la requête persiste, c'est pourquoi il est nécessaire que l'utilisateur d'un tel système joue aussi un rôle d'acteur sur le résultat obtenu par le système.

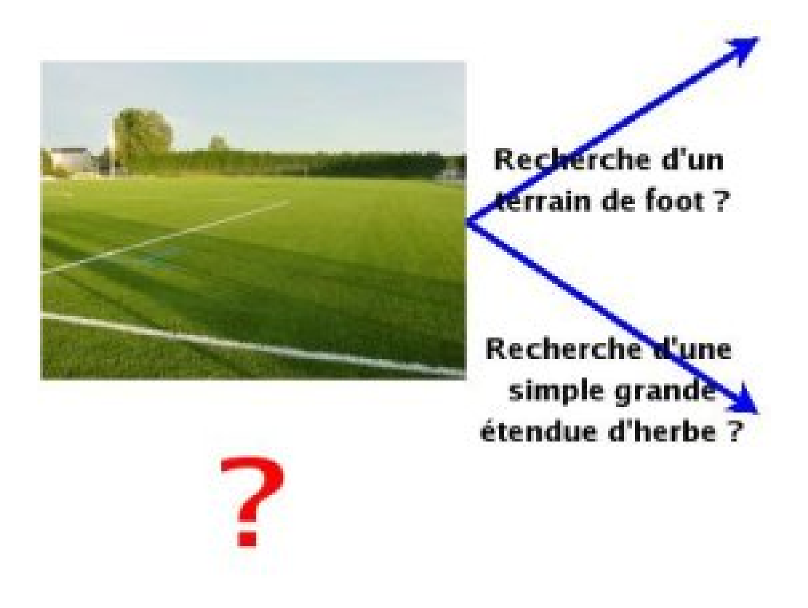


FIG. 3.5 – Exemple de subjectivité de la recherche picturale

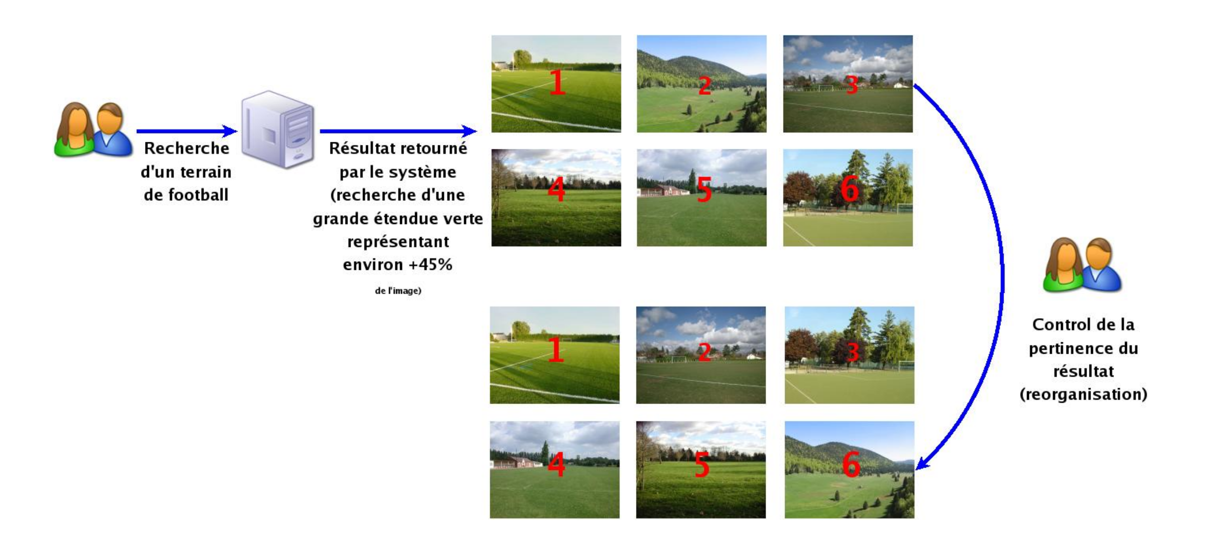


FIG. 3.6 – Exemple de contrôle de pertinence (Relevant Feedback)

3.2 Indexation de l'information

Une fois les informations de l'image représentées sous forme de signatures, il est nécessaire de réfléchir aux problématiques liées à l'organisation, le stockage et la recherche de celles-ci ; c'est-à-dire comment procéder à l'indexation de l'information. Il faut bien comprendre le caractère restrictif des signatures aussi pertinentes soient elles. En effet, la signature contient bien évidemment beaucoup moins d'informations que la zone de l'image sur laquelle elle a été effectuée puisqu'elle se veut concise et représentative d'un sous-ensemble d'informations par rapport à l'ensemble des informations d'origine relative à la zone de l'image sélectionnée. Il faut donc garder à l'esprit qu'il est nécessaire de conserver avec la signature un ensemble minimal d'informations de la zone sur laquelle celle-ci s'est appliquée. Il est donc intéressant de mémoriser, quand cela est possible, les coordonnées relatives de la zone de l'image signée ou encore la couche signée ainsi que sa signification (par ex. : un label ou un nom de couche) et enfin les relations avec les autres « éléments » signés.

L'indexation d'une image quant à elle, consiste à repérer dans celle-ci certaines informations particulièrement significatives (ce que nous appelons ici descripteurs ou signatures) dans un contexte le plus général possible, et à créer un lien entre ces descripteurs et l'image originale. Par exemple, en bibliographie, les pages d'index d'un livre reprennent (parfois) les termes significatifs apparaissant dans celui-ci, et les relient aux pages du livre où ces termes (ou leurs synonymes) apparaissent. Ceci facilite pour le lecteur la localisation des pages ou sections où l'on mentionne un sujet particulier. De même, la table des matières d'un livre est une forme (assez grossière) d'indexation.

Revenons à notre contexte pictural et tentons de lui appliquer un exemple concret. Imaginons le type de requête suivante : « rechercher toutes les images contenant un ballon de volley ». Il faut donc pouvoir retrouver toutes les occurences de ballons (et notamment de volley) dans l'ensemble des données et renvoyer les images auxquelles



elles sont associées. Pour cela, il est recommandé d'organiser les données à la manière d'une table des matières ou « index » où chaque descripteur est associé à l'image qu'il décrit afin que la recherche soit la plus rapide possible.

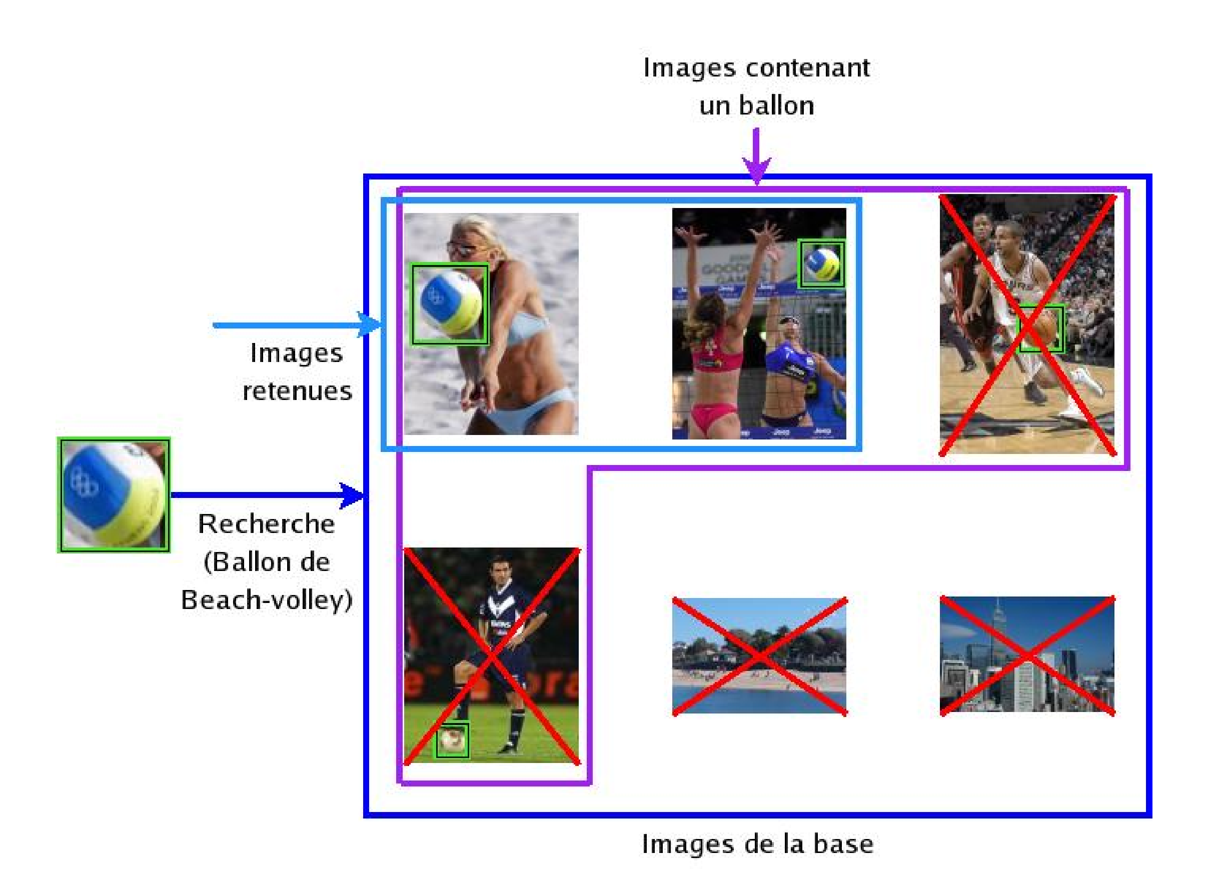


FIG. 3.7 – Exemple de recherche dans une base d'images

L'indexation peut être manuelle (faite par un humain), automatique (créée par un programme informatique), ou à divers degrés intermédiaires « assistée » ou semi-automatique (par exemple créée par un humain assisté d'un programme proposant des informations-clés). L'indexation manuelle d'informations est généralement coûteuse : pour indexer correctement des documents picturaux anciens, il faut faire intervenir des personnes qui soient elles-mêmes capables de comprendre le contenu du texte, ce qui impose un coût non négligeable.

Bien que l'indexation se base sur des techniques relativement établies, il peut y avoir plusieurs indexations différentes d'un même document, aussi valables les unes que les autres, en fonction de l'usage qui doit en être fait et du public auquel elles s'adressent. À titre d'exemple, imaginez une image qui soit une description d'une lettrine d'un document ancien du XVII ^{e} siècle sous Louis XIV ; son indexation sera très différente selon que le public sera constitué d'historiens ou de graphistes.



En ce qui concerne les relations entre les différents « éléments » signés, il est intéressant de revenir sur le cas concret que l'image de la joueuse de beach-volley. Nous avons vu précédemment, que l'on pouvait par exemple signer l'image dans sa globalité puis ensuite signer un élément qui la compose comme le personnage principal à savoir la joueuse et enfin son visage. On observe nettement que nous avons une imbrication d' « éléments » signés. Il peut donc exister une relation de hiérarchie entre les « éléments » de l'image que l'on peut représenter par un graphe et plus précisément un arbre. Voilà pourquoi, il est véritablement pertinent de conserver cette information en plus des signatures calculées.

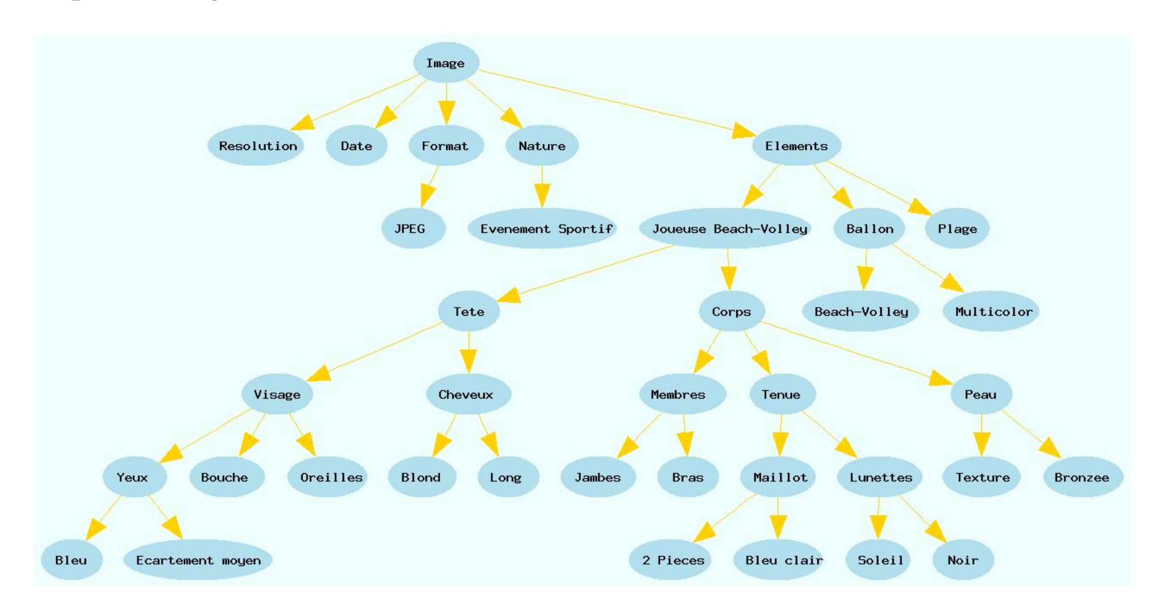


FIG. 3.8 – Exemple de graphe des informations d'une image

Il en résulte alors un ensemble de problématiques, comme la construction de ces arbres d'informations et leurs manipulations.

3.3 Besoins et contraintes du fonctionnement de la plate-forme

Comme énoncé dans la présentation du sujet, l'application permettant d'effectuer la recherche et l'indexation des images doit être une application de type plate-forme Web, ceci afin d'obtenir une plate-forme simple d'utilisation et utilisable depuis n'importe où sur le réseau avec un navigateur internet.

3.3.1 Utilisateurs de l'application

Comme il a été précisé dans le sujet, l'application devra pouvoir s'adresser à des utilisateurs non qualifiés que ce soit dans le domaine informatique ou encore de l'imagerie. L'interface mais également le fonctionnement global devront donc prendre en compte cette exigence afin de simplifier au mieux l'utilisation de la plate-forme.



3.3.2 Sécurité

L'application devant manipuler à terme une base de données qui peut s'avérer importante, il est important de sécuriser l'accès à cette dernière ainsi que les données manipulées, d'autant plus que cette application peut-être accessible depuis le réseau des réseaux.

3.3.3 Réactivité

Bien que l'application devra être capable d'effectuer de lourdes opérations comme le calcul de signatures, et l'indexation des informations de l'image, la notion de vivacité du système est une notion qui conserve son importance dans le développement de la plate-forme. L'utilisateur final ne devrait pas avoir l'impression que le système se bloque lors de son exécution et devrait pouvoir obtenir le résultat de ses requêtes dans un délai minimum.

3.3.4 Interopérabilité

Le calcul de descripteurs d'images est actuellement le fruit du travail d'un grand nombre d'acteurs du consortium. L'application ayant également pour but de s'interconnecter avec le travail de ceux-ci, elle devra garantir la possibilité d'intégrer facilement différents modules externes dont l'objet est le calcul d'un ou plusieurs descripteurs. Une méthodologie en réponse à ses attentes sera alors à établir.

3.3.5 Déploiement et portabilité

Le système devra pouvoir s'installer facilement et être utilisable sur un maximum de plate-formes que ce soit au niveau de l'utilisateur ou encore au niveau de l'administrateur du système. Des mesures devront donc être prises dès le départ du développement afin que ces contraintes soient prises en compte.

3.3.6 Contraintes d'une plate-forme web

Le développement d'applications basées sur une architecture de type « application web » engendre bien souvent des contraintes d'ordre technique. Il est donc essentiel de les dénombrer et d'effectuer des choix sur les compromis qu'elles engendreront.

Deuxième partie

Étude des problématiques de développement de la plateforme

Chapitre 4

Génération, manipulation et visualisation de l'information

4.1 De l'image à l'information « utile »

Nous avons vu précédemment (cf. Sous-Partie 3.1, page 11) la grande quantité d'informations que l'on peut extraire d'une image. Certaines sont subjectives et devront être renseignées par l'utilisateur, d'autres peuvent être calculées à partir du contenu numérique qu'offre un fichier représentant l'image.

Dans un fichier, on trouve généralement deux catégories d'informations que l'on peut extraire :

1. Les informations de numérisation de l'image (format, résolution, compression, colorimétrie, poids du fichier, date de création etc...) et méta-données du format de fichier si elles existent (résolution de numérisation (souvent exprimée en dpi [1]), appareil utilisé pour la numérisation, texte supplémentaire d'annotation de l'image, etc...)
2. Les pixels de l'image

Pour la première catégorie, nul besoin de traitement, l'information est aussitôt exploitable ; par contre la seconde nécessite des traitements pour extraire l'information utile, les différents « descripteurs » (ou signatures cf. Sous-partie 3.1.1, page 13) de l'image.

De nos jours, de nombreux outils permettent de manipuler le contenu d'une image et notamment les pixels de celle-ci. Cependant, dans le cadre de ce projet, l'outil doit répondre à des besoins bien spécifiques, c'est-à-dire être capable d'effectuer n'importe quel type de calcul sur les pixels. Le type de solution qui répond le mieux à cette attente est bien évidemment d'utiliser une bibliothèque pour un langage de programmation lambda permettant ainsi de répondre à la plupart de nos attentes, d'autant plus que de nombreuses bibliothèques de ce genre existent pour presque tous les langages de programmation. Il suffira alors d'implémenter les algorithmes de calcul dans ce langage en utilisant la bibliothèque afin d'obtenir les informations - « descripteurs » - voulues.

De plus, l'intérêt d'utiliser une bibliothèque graphique associée à un langage est

[1] unité mesurant la résolution d'un scanner - on parle de « finesse de numérisation ». Plus cette valeur est élevée et meilleure est la qualité.

double puisque cela peut rendre le module ainsi créé réutilisable dans d'autres domaines moyennant certaines contraintes liées au langage et/ou à la bibliothèque associée.

4.2 De l'information « utile » à la base de données

Une fois toutes les informations obtenues, elles devront être organisées et enregistrées dans un conteneur. Précédemment (cf. Sous-partie 3.2, page 16), les problématiques liées à l'organisation des informations et plus précisément les liens qu'il peut y avoir entre elles, ont été cités et l'idée de les organiser sous forme d'arbres a été soumise.

De plus, conformément à ce qui est demandé dans le cadre de ce stage (cf. Sous-partie 2.2, page 10), il est a priori logique, du moins dans un premier temps, d'organiser les informations dans un arbre au format XML [2], bien que d'autres solutions auraient pu être possibles mais pas forcément pertinentes (format de texte simple, structure tabulaire semi structurée, etc...).

4.2.1 Le XML

XML est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage visant à mettre en forme des données grâce à des balises (markup).

Contrairement à HTML, qui est à considéré comme un langage défini et figé (avec un nombre de balises limité), XML peut être considéré comme un méta-langage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises décrivant l'organisation de données.

La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

XML a été mis au point par le XML Working Group sous l'égide du World Wide Web Consortium (W3C) dès 1996. Depuis le 10 février 1998, les spécifications XML 1.0 ont été reconnues comme recommandations par le W3C, ce qui en fait un langage reconnu. (Tous les documents liés à la norme XML sont consultables et téléchargeables sur le site web du W3C, http://www.w3.org/XML/)

XML est un sous ensemble de SGML [3], défini par le standard ISO8879 en 1986, utilisé dans le milieu de la Gestion Électronique Documentaire (GED). XML reprend la majeure partie des fonctionnalités de SGML. Il s'agit donc d'une simplification de SGML afin de le rendre utilisable sur le web !

[2]eXtensible Markup Language que l'on peut traduire par langage étendu à balises, ou langage extensible à balises

[3]Standard Generalized Markup Language



Les avantages de XML

Voici les principaux atouts de XML :

- La lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML.
- Auto-descriptif et extensible.
- Une structure arborescente : permettant de modéliser la majorité des problèmes.
- Universalité et portabilité : les différents jeux de caractères sont pris en compte.
- Déployable : il peut être facilement distribué par n'importe quel protocole à même de transporter du texte, comme HTTP.
- Intégrabilité : un document XML est utilisable par toute application pourvue d'un parser [4].
- Extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications.

Ainsi, XML est particulièrement adapté à notre problème puisqu'il permet de conserver la structure arborescente de nos données, qu'il permet de s'intégrer à tout type d'outils et d'être manipulé au sein d'une application web puisque distribuable par le protocole HTTP.

Fonctionnement de XML

La norme XML en tant que telle doit être vue comme un outil permettant de définir un langage (on dit alors qu'il s'agit d'un métalangage), permettant de créer des documents structurés à l'aide de balises.

Une balise est une chaîne de caractères du type :

```
<balise>
```

Ainsi, un document XML créé en suivant les spécifications de la norme XML pourra par exemple ressembler à ceci :

```
<image>
<file>
        <filename>french_country</filename>
        <filetype>JPEG</filetype>
        <filesize>20480</filesize>
</file>
<metadata>
        <creation_date>20050123</creation_date>
        <creator>John Smith</creator>
        <type>landscape</type>
</metadata>
</image>
```

La syntaxe des éléments en XML

L'arbre des éléments, c'est-à-dire le véritable contenu du document XML, est constitué d'une hiérarchie de balises comportant éventuellement des attributs.

[4] c'est-à-dire un logiciel permettant d'analyser un code XML

Un attribut est une paire clé valeur écrit sous la forme Clé="Valeur", ainsi une balise affectée d'un attribut aura la syntaxe suivante :

```
<balise clé="valeur">
```

Toute donnée est ainsi encapsulée entre une balise ouvrante *<balise>* et une balise fermante *</balise>* (Sachant qu'une donnée peut éventuellement être un ensemble d'éléments XML). Ainsi, un élément vide est uniquement constitué d'une balise spécifique dont la syntaxe est la suivante : *<balise/>*.
D'autre part, il est interdit en XML de faire chevaucher des balises, c'est-à-dire d'avoir une succession de balises du type :

```
<balise1>

<balise2>

</balise1>

</balise2>
```

Autres perspectives au sein de XML

Précédemment, XML a été présenté comme un outil permettant de définir un métalangage pouvant apporter des solutions dans un contexte précis. On peut donc utiliser XML en définissant, selon nos propres besoins, un langage permettant de décrire et organiser nos données dans le contexte de documents visuels comme les images. Cependant, avant de vouloir tout définir, il faut se demander si le problème n'a pas déjà été posé dans la communauté informatique.

En effet, la question est loin d'être nouvelle et un ensemble d'acteurs travaille actuellement sur le sujet afin de définir un standard tentant de répondre à un certain nombre de questions sur la description de contenu multimédia incluant les documents de type « images ».

4.2.2 Norme MPEG-7

Actuellement, aucun processus standard ne permet de retrouver, à la manière des moteurs de recherches textuelles, des contenus multimédias.
Afin de pallier ce problème, des chercheurs du Moving Picture Experts Group (MPEG) ont développé un nouveau standard ISO sous le nom de MPEG-7. Pour les initiateurs de ce dernier, « la principale ambition de MPEG-7 est de rendre les informations multimédias aussi faciles à trouver sur le Web que le texte l'est aujourd'hui ».

Bien que cette appellation soit dans la même lignée que les standards MPEG-1, 2 ou 4, il ne s'agit pas là d'un nouvel algorithme de compression de données vidéo, mais d'un système d'encodage de description de contenu appelé « Multimedia Content Description Interface ».

Chaque description encodée est associée à un contenu précis d'une partie d'une vi-



déo, d'une image ou d'un son. Différents niveaux permettent, par exemple, d'obtenir des informations sur :
- la forme, la taille ou la texture d'une image ;
- le type de scène et les objets/acteurs figurant dans cette scène ;
- le type de son, le copyright et le prix du morceau.

Plus simplement, la norme MPEG-7 décrit les caractéristiques des différents contenus multimédias de façon à pouvoir, par la suite, rechercher et extraire tout ou partie d'un contenu d'une vidéo, d'un son ou d'une image.

Techniquement, deux éléments principaux font partie du standard MPEG-7. Il s'agit :
- de la description du langage (DDL - Description Definition Language) qui définit la syntaxe pour la création des descriptions ;
- de l'outil ou schéma de description (DS - Description Schemes ou Description Tools) qui définit la sémantique de chaque donnée ou méta donnée.

« DDL » fait partie du noyau de la norme MPEG-7. Elle fournit la base par laquelle les fournisseurs peuvent créer leurs descriptions selon les règles syntaxiques définies. Plus précisément, il s'agit d'un langage qui permet la création de nouvelles données basées sur la syntaxe XML.

Toutefois, comme le langage XML n'a pas été développé spécifiquement pour des contenus multimédias, un certain nombre d'extensions a été rajouté à celui-ci, spécifiquement pour la norme MPEG-7.

Le schéma de description (DS) est la structure des méta-donnés pour décrire et annoter des contenus audiovisuels (AV). Le « DS » fournit une manière standardisée pour décrire la gestion et la description du contenu afin de faciliter la recherche, le classement et l'accès aux informations. La gestion et la description sont définies en utilisant le langage « DDL ». Ces deux informations peuvent être exprimées en forme textuelle (lisible par l'humain) ou binaire comprimée (prévu pour le stockage ou pour la transmission).

Plus concrètement, lors de la création d'un contenu MPEG-7, par exemple pour une image, un minimum de quatre étapes d'enregistrement d'informations doit être pris en considération :

1. l'information sur la création du contenu et la description du média utilisé et utilisable ;
2. la forme de l'image principale, sa texture et sa couleur ;
3. la forme d'un ou de plusieurs détails dans l'image comprenant également leurs textures et leurs couleurs ;
4. la couleur, la texture et l'histogramme de l'image de fond.



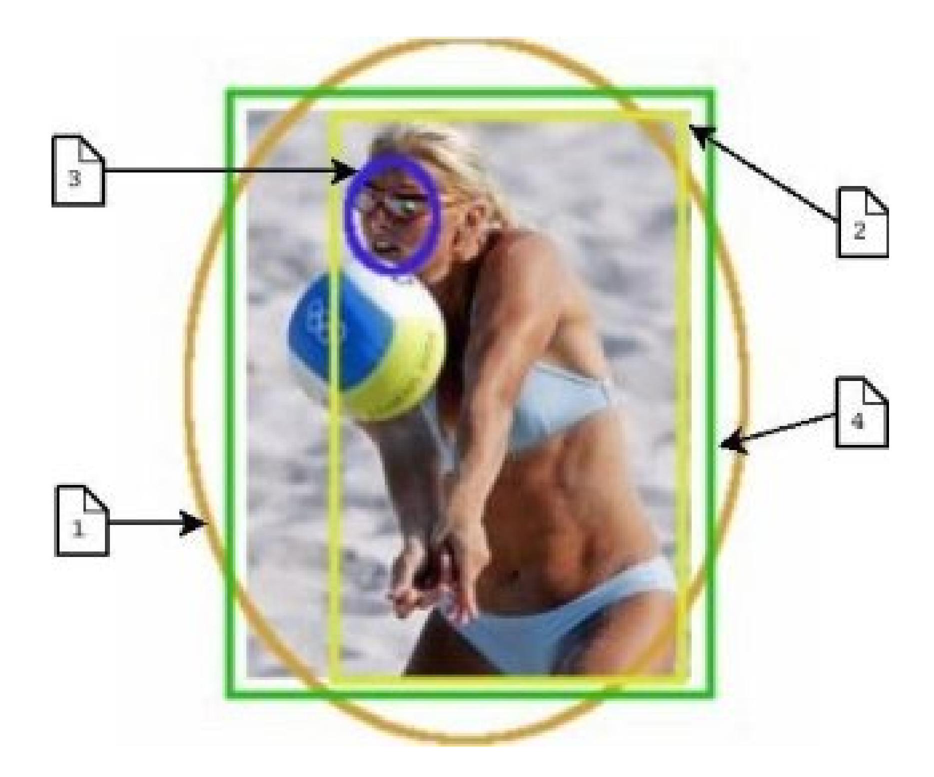


FIG. 4.1 – Exemple d'extraction d'informations utilisées dans le MPEG-7

Partant de là, de nombreuses applications sont envisageables avec la norme MPEG-7. Les exemples pour ce type d'applications sont :

- pour la vidéo
  - l'accès à l'information du type de montage ou de la structure de la vidéo ;
  - l'accès à des scènes précises par des recherches de sons, phrases, musiques ou par une description précise d'un objet, d'une marque ou d'un acteur ;
  - l'accès à des segments de vidéos basés sur des critères subjectifs plutôt qu'objectifs.
- pour l'audio
  - l'accès à un morceau précis selon une mélodie sifflée, chantée ou jouée ;
  - l'identification d'un morceau passé à la radio ou diffusé à partir d'un support audio ;
  - la recherche du texte d'une chanson d'après les premiers airs entonnés.
- pour l'imagerie
  - la recherche d'images ou de photos selon un certain type de logo, de visage, de texture ou de couleur.

Si cette méthode de recherche présente de nombreux avantages, son inconvénient majeur est le volume que risquent de prendre toutes ces nouvelles informations insérées en plus du contenu de base.

A l'inverse, au travers de ces nouvelles données, l'utilisateur pourra définir ses critères de sélection et de recherche dépendant de son interface d'affichage (PC, téléphone mobile, PDA, notebook, etc...) et de connexion (xDSL, câble, GSM, UMTS, WLAN, etc...).



Bien que la norme MPEG-7 soit standardisée depuis quelques temps, très peu d'applications sont encore disponibles car il s'agit avant tout de créer de nouveaux contenus. D'autre part, depuis peu, un nouveau standard a vu le jour sous le nom de « MPEG-21 », 21 correspondant probablement au 21e siècle, qui a pour but non pas de remplacer MPEG-7, mais de proposer une architecture unifiée pour la définition, le traitement et la recherche de contenu multimédia, appelée « UMA » (Universal Multimedia Access).

L'objectif principal de cette nouvelle norme est de standardiser une structure dite « entité numérique » ou Digital Item, dans son langage de base, faisant référence à n'importe quel contenu vidéo, image, texte, méta-donnée ou son.

Avantages et inconvénients pour notre projet

On observe clairement l'avantage de ce type de norme pour la description de nos images, car il est d'autant plus complet et exhaustif mais est du surcroît une norme de description, - **un standard**. Cependant, il ne faut pas oublier qu'autant de règles à respecter peuvent s'avérer être d'une grande « lourdeur » au niveau du développement d'applications de part les contraintes à respecter et le manque de flexibilité que le standard peut imposer. Le MPEG-7 n'étant qu'à ses débuts, le manque de documentation et sa vitesse d'évolution peuvent s'avérer être un réel handicap dans notre projet.

Dans notre contexte de développement, il serait plus raisonnable dans un premier temps, de s'inspirer de cette norme plutôt que de la suivre à la lettre, afin de garder une certaine flexibilité et moins de contraintes quitte à s'en rapprocher dans le futur dans un souci de standardisation.

4.2.3 Type de base de données et SGBD

Un ensemble d'informations constitue ce qu'on appelle habituellement une base de données. Une base de données constitue en réalité un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données). Une base de données se traduit physiquement par un ensemble de fichiers sur disque. On dénombre actuellement plusieurs types de bases de données :
– Les bases de données dites « plates » : Les données sont stockées dans un fichier, ou chaque ligne représente un élément (n-uplet) de la base.
– Les bases de données hiérarchiques : Les données sont structurées dans des hiérarchies, comparables à l'organisation des répertoires sur un ordinateur.
– Les bases de données réseaux (extension du modèle hiérarchique).
– Les bases de données relationnelles : des données hétérogènes sont stockées dans des tables, permettant d'établir des relations entre elles.
– Les bases de données orientées objet : dans une telle organisation, les données sont représentées sous forme d' « objets ». Ceux-ci contiennent les données qui les décrivent et qui représentent leur « état ».
– Les bases de données semi-structurées (ex. : bases XML).



Le choix du corpus implique bien évidement des conséquences sur le système qui sera utilisé pour le manipuler, à savoir le SGBD ou système de gestion de base de données.

Le dilemme entre le relationnel et le XML pur

Nous avons établi précédemment (cf. Sous-partie 4.2.2, page 27) la méthode selon laquelle seront organisées nos informations relatives à nos images. Cependant, lors d'un long débat au sein de l'unité d'enseignement Vision 2 [5], il a été démontré qu'il était possible de représenter l'intégralité de l'information d'une image en utilisant un ensemble de tables ayant des relations entre elles, que ces relations soient de type hiérarchique ou n-aire.

On pourrait donc utiliser soit une base de données relationnelle, soit un système semi-structuré en utilisant une base de données XML.

Toutefois, il est bien évidemment plus naturel, à partir de données organisées sous le format XML, d'utiliser une base forestière [6] plutôt qu'une base relationnelle dénaturant l'organisation d'origine, pour insérer les données dans des tables.

Il existe actuellement de nombreux outils pour la manipulation de bases qu'elles soient relationnelles ou XML. En effet, les équivalences sont nombreuses que ce soit au niveau du langage de requêtes SQL pour le relationnel ou XQuery/XPath pour le XML, ou encore pour l'implémentation du SGBD : Oracle, MySQL, PostgreSQL ou eXist, dbXML et Apache Xindice.

4.3 Visualiser et utiliser les données

A partir d'un ensemble d'images, le système doit être capable d'indexer et d'effectuer, par exemple, des requêtes de type sélection sur celles-ci et de rendre le résultat affichable sur une simple page internet. Avant d'effectuer tout type de développement internet, il faut au préalable choisir le type de plate-forme sur laquelle reposera l' application. Une application web ne peut bien évidemment fonctionner seule. Elle a besoin d'au moins une machine appelée « serveur Web » sur laquelle s'exécutera en permanence un « logiciel » ou démon appelé HTTPd dont la charge sera d'interpréter et d'envoyer aux différents navigateurs internet clients des pages HTML. Pour cela, nous devons tout d'abord définir au moins deux éléments :
- le type de la ou les machines physiques sur laquelle ou lesquelles s'exécutera le démon HTTPd
- le démon HTTPd lui-même

4.3.1 Déterminer le type de « solution web »

Il existe à l'heure actuelle quatre solutions pour le développement d'applications web basées sur le démon HTTP de quatre grands groupes internationaux :

[5] Unité d'enseignement de 2ième année du Master 2 IMA à l'université de la Rochelle
[6] base constituée d'une ensemble d'arbres XML dit forêt



– Solution basée sur le serveur HTTP Apache de la Apache Software Foundation, successeur du NCSA httpd
– Solution basée sur le serveur HTTP Internet Information Services de Microsoft
– Solution basée sur le serveur HTTP Sun ONE de Sun Microsystems (anciennement iPlanet de Netscape)
– Solution basée sur le serveur HTTP Zeus de Zeus Technology (http://www.zeus.com/)

Il convient donc d'effectuer la comparaison de ces quatre solutions afin d'en ressortir la solution qui correspond le mieux aux besoins du projet.

Voici donc un petit tableau récapitulatif des différents avantages et inconvénients de ces quatre solutions :

Nom du produit	Apache	IIS	Zeus	Sun One
Points forts	- Gratuité - Peu gourmand en ressources matérielles - Excellente stabilité	- Grande simplicité d'installation et d'utilisation - Peu gourmand en ressources matérielles	- Excellente stabilité - Capacité à tenir de très gros pics de fréquentation - Interface native avec certaines bases de données	- Isolation de processus - Restauration des données de configuration
Points faibles	- Installation et administration plus laborieuses (pas d'interface graphique) - Pas de support technique	- Coût du support technique - Très gourmand en ressources matérielles	- Coût de la licence (environ 1700 €) - Coût du support technique	- Coût de la licence - Très gourmand en ressources matérielles

TAB. 4.1 – Avantages et inconvénients des différentes solutions web

Il est également primordial de comparer plus en détail, les coûts de toutes ces solutions mais également les environnements supportés :

Nom du produit	Apache	IIS	Zeus	Sun One
Prix	- 0 €	- inclus dans Windows 2000 serveur, lequel vaut environ 1300 €	- environ 1900 €	- environ 1650 €/processeur
Compatibilité	- Linux, Windows 98 à XP, de nombreux Unix, MacOS X	- Windows 2000 Server	- Linux, de nombreux Unix, MacOS X	- Solaris, Linux Red Hat Advanced Server, Windows 2000 & 2003 Server et HP-UX

TAB. 4.2 – Coût et compatibilité des différentes solutions web



FIG. 4.2 – Solution Web complète proposée par Microsoft

Rappelons que le projet a pour client le laboratoire L3i, le coût de la solution est donc un facteur déterminant vu les moyens limités du secteur de la recherche. Outre le coût, le développement sur Apache bénéficie d'une communauté importante et de nombreuses documentations disponibles sur Internet, réduisant ainsi la difficulté toute relative de l'installation et de l'administration du serveur. Enfin, c'est l'offre la plus complète en ce qui concerne la panoplie d'environnement de déploiement.



Ensuite, il est évident que des pages HTML écrites statiquement seraient incapables d'effectuer ce qu'on attend de cette application. Les pages seront donc générées de façon dynamique par le biais d'un langage de scripts comme PHP, Perl ou Python. La solution retenue sera donc une solution de type LAMP combinée à un certain nombre de modules répondant à nos besoins spécifiques.

La plate-forme LAMP



FIG. 4.3 – LAMP : Solution Web complète et Open Source

LAMP est un nom usuel pour désigner les plates-formes combinant :
- le système d'exploitation GNU/Linux
- serveur Web Apache
- système de bases de données MySQL
- et bien sûr PHP, Python ou Perl

Par extension, sous environnement Windows, on parlera de plate-forme WAMP. Trois WAMP « intégrés » particulièrement connus sont EasyPHP, Xampp et WAMP55.

Avantages des technologies Open Source :

L'utilisation des technologies Open Source confère à ses utilisateurs de nombreux avantages :
- Stabilité et robustesse
- Inter-opérabilité des logiciels
- Assistance et support d'une communauté de plusieurs milliers de développeurs
- Code source disponible
- Conformité aux normes et standards
- Adapté à des environnements productifs industriels
- Performance et fiabilité

4.3.2 Besoins en terme d'interface

Cette application doit être utilisable par un ensemble assez large d'utilisateurs (cf. Sous-partie 3.3.1, page 18) et doit ainsi être assez simple et agréable d'utilisation. Il faut donc simplifier au maximum le nombre d'interventions en terme d'actions sur l'interface ainsi que le nombre de paramètres à saisir ou à modifier.

Pour cela, nous devons définir clairement les types d'utilisateurs et les différents cas d'utilisation qui leur sont associés. Voici le schéma représentant l'ensemble des utilisateurs associés à leurs actions possibles :

Effectuer une recherche dans la base

Utilisateurs standards

Mettre à jour l'ensemble les calculs de descripteurs d'images de la base

Administrateur

Insérer de nouvelles images dans la base

Développeurs

Insérer de nouveaux outils de calcul de descripteurs d'images dans la base

FIG. 4.4 – Diagramme des cas d'utilisation de la plate-forme web

A partir de là, nous pouvons élaborer des maquettes d'interface qui répondrons à ces différents cas. Le cas d'utilisation le plus intéressant et celui qui sera le plus usité et sans doute le cas de la recherche d'images semblables à une image de référence. Il peut se résumer à l'ensemble d'actions suivant :

1. Sélectionner une image requête
2. Définir la portée de la recherche (action facultative si on admet la possibilité d'un comportement par défaut)
3. Lancer la recherche

L'interface de ce cas d'utilisation devra donc se schématiser de la façon suivante :



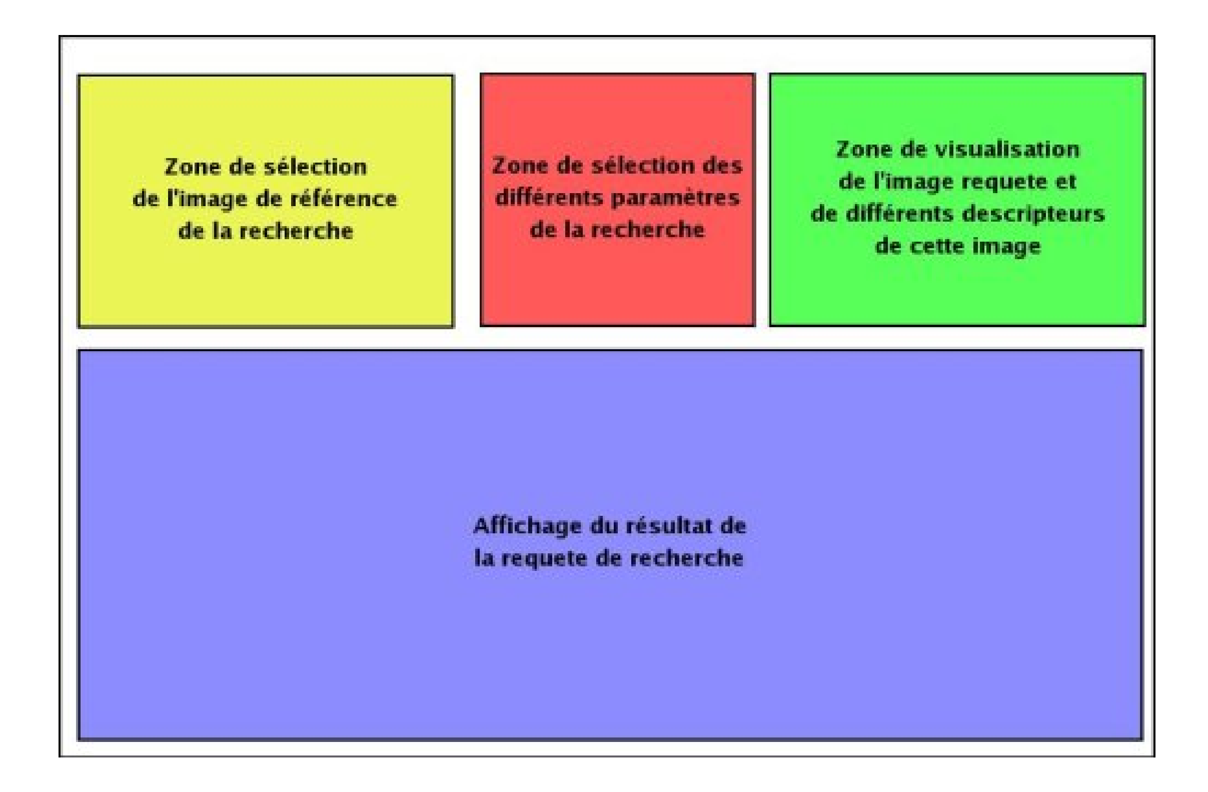


FIG. 4.5 – Maquette de la page de recherche de l'application

L'important dans cette action est de réduire au maximum le nombre de paramètres à préciser sans diminuer les possibilité de l'application. Les paramètres intéressants pourraient par exemple se limiter à ceux-ci :

– Nombre de résultats à retourner
– Nature de l'information à rechercher

Cependant, le terme de « Nature de l'information » a besoin d'être éclairci. Nous avons vu précédemment (cf. Sous-partie 3.1.1, page 13) que la quasi totalité de l'information réelle de l'image pouvait être représentée par un ensemble de signatures ou descripteurs de l'image. On pourrait donc associer en quelque sorte à chaque descripteur ou ensemble de descripteurs, un « label » qui représenterait la nature de l'information que cet ensemble décrit. Ainsi, l'utilisateur pourrait sélectionner à sa guise le type de recherche qu'il souhaite effectuer en cochant les différents types d'informations pertinentes pour sa recherche.

4.3.3 Contraintes au niveau de la sécurité de l'application

Afin d'éviter toute éventuelle détérioration des données et dans un souci de restriction concernant l'accès à l'application, des mesures de sécurité seront prises. Premièrement, l'application étant accessible depuis internet, l'accès au serveur web (sécurité physique de la machine, ainsi qu'une configuration du démon HTTPd depuis les paramètres de Apache) devra donc être surveillé et limité. Ensuite, il faudra bien évidemment veiller aux droits associés à l'arborescence contenant les différentes bases de données de l'application. Enfin, afin de restreindre l'accès de l'application aux seuls utilisateurs concernés, une méthode d'authentification sera mise en place parallèle-

ment à la création informatique d'utilisateurs.

4.3.4 Vivacité de la plate-forme

Des besoins de réactivité concernant l'application ont été exprimés (cf. Sous-partie 3.3.3, page 19) afin de rendre l'application utilisable et conviviale dans un navigateur internet. Cela n'est en général pas un souci pour la plupart des applications web, sauf lorsqu'il s'agit d'effectuer des calculs lourds comme c'est ici le cas (calcul de descripteurs sur une ou plusieurs images). Il existe différentes approches possibles et combinables pour résoudre ce problème :

- Utiliser une machine suffisamment puissante pour effectuer rapidement tous les calculs demandés (études des besoins en terme de puissance de calcul et en terme de mémoire nécessaire)
- Séparer les différentes parties de l'application (calcul d'un côté et le reste de l'autre) sur des machines distinctes (Utilisation d'un serveur web ET d'un serveur de calcul)
- Diviser le calcul sur plusieurs machines afin de répartir la charge
- etc...

La première solution n'est évidemment pas une des meilleures puisqu'il est difficile d'évaluer la puissance et la mémoire demandées d'autant plus que si les besoins tendent à évoluer, la solution devient coûteuse puisqu'il est ensuite nécessaire d'améliorer la machine, voire de la changer.

Les seconde et troisième solutions sont évidemment meilleures, car elles permettent de ne pas surcharger le serveur web en terme de charge mais demande une politique de mise en place particulière, et encore plus pour la dernière. On pourra dans un premier temps séparer la partie calcul sur une machine dédiée et ensuite décider d'ajouter un intermédiaire qui s'occupera de distribuer les calculs à d'autres machines rendant le processus transparent pour le serveur web.



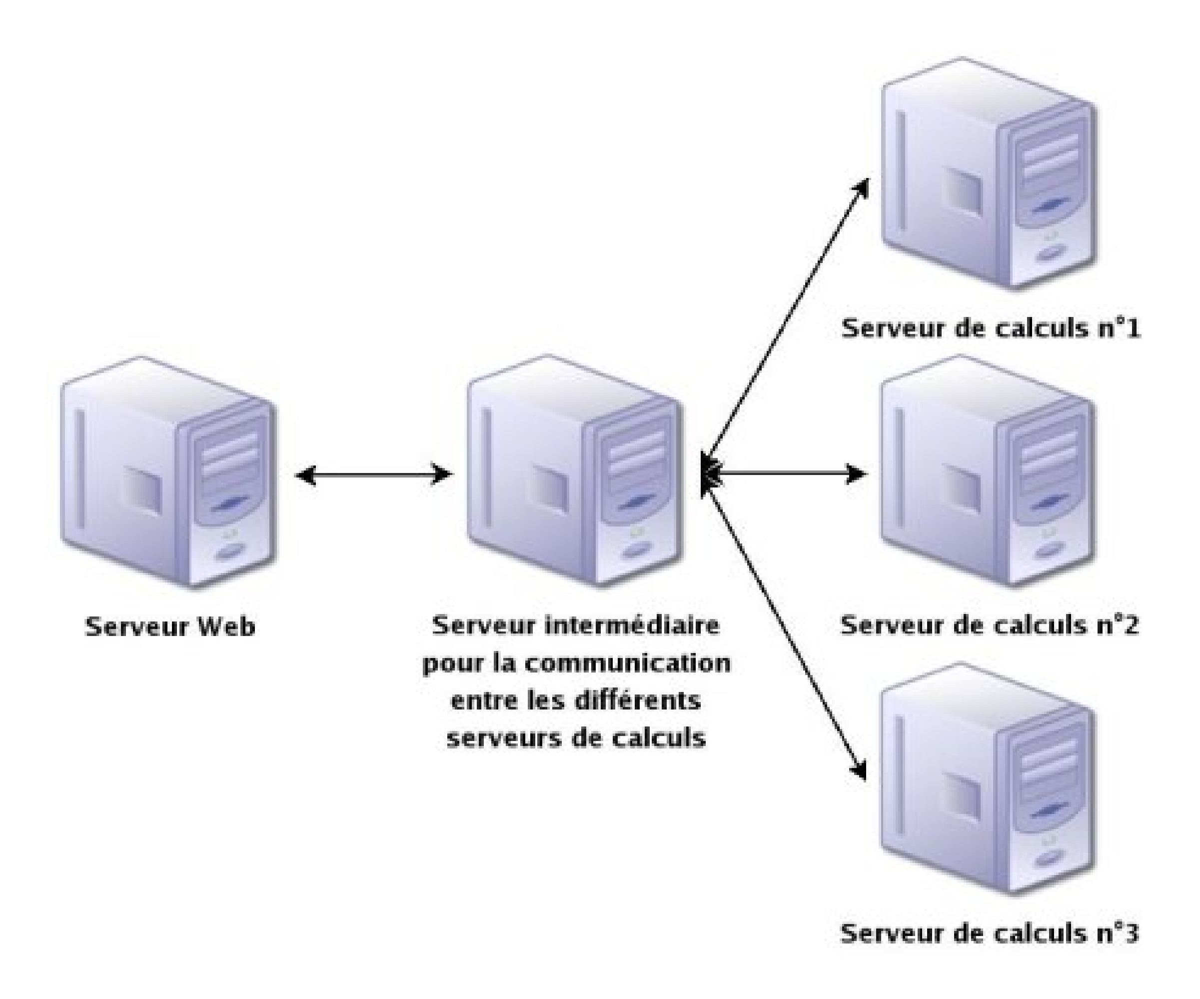


FIG. 4.6 – Schéma de répartition des calculs avec serveur intermédiaire

4.3.5 Restrictions et Compromis

Dans la partie relative aux problématiques liées aux besoins et aux contraintes (cf. Sous-partie 3.3.6, page 19), nous avons fait ressortir qu'il pouvait y avoir des compromis à effectuer étant donné la nature « web » de l'application. Nous avons également souligné l'interopérabilité (cf. Sous-partie 3.3.4, page 19) que l'application devait avoir.

En effet, l'application devant pouvoir exécuter et traiter les résultats de tout « programme » écrit par d'autres collaborateurs du consortium Madonne, et ce sur une plate-forme web, il devient nécessaire de nuancer cette contrainte en regard aux limitations dues à l'architecture web.

Pour rappel, un serveur web interprète une suite d'instructions afin de générer des pages HTML sur le navigateur internet du client. Afin que le client n'attende pas indéfiniment l'interprétation du serveur Web, des limitations d'ordre mémoire et temps d'exécution d'un script sont fixées. Ainsi, si un script demande trop de mémoire ou un temps d'exécution trop long, il sera tout simplement stoppé. De ce fait, pour obtenir des résultats corrects mais néanmoins rapides, les temps d'exécution devront rester les plus courts possibles.



Le fait que les calculs soient exécutés en dehors du serveur web ne change pas la donne (excepté pour la partie mémoire) puisque le temps continue de s'écouler entre le moment où le serveur web effectue sa demande auprès du serveur de calcul (ou serveur intermédiaire) et le moment où le serveur de calcul a fini son exécution et renvoie son résultat. Tout le temps nécessaire pendant et entre ces étapes est inclu dans le temps d'exécution global du script puisque le serveur web reste en attente du résultat afin de continuer son exécution.

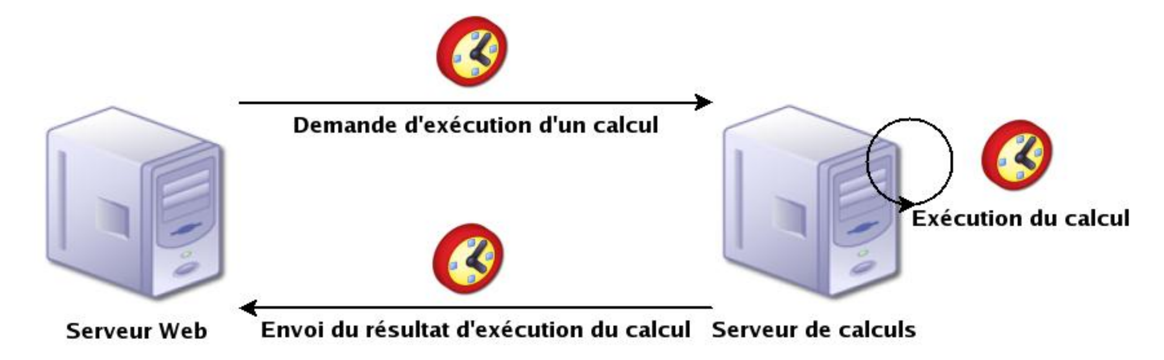


FIG. 4.7 – Schéma d'exécution d'un calcul dans l'application

Enfin, afin que le serveur de calcul puisse exécuter un programme développé par un collaborateur du consortium Madonne, il est nécessaire de normaliser le comportement de l'ensemble des programmes pouvant être interfacés. En effet, il serait impossible que la plate-forme puisse s'interfacer avec des modules totalement hétérogènes. Afin que l'interopérabilité soit conservée, le minimum de conditions à respecter est de garantir que tous les modules aient le même type de données en entrée et le même type de données en sortie.

Ainsi, les modules ou « plugins » seront des programmes compilés (exécutables sur n'importe quel système d'exploitation, afin de ne pas nuire à la portabilité de l'intégralité du système) et livrés avec toute leur dépendance de bibliothèques associées (alors des sources compilables avec le manuel d'installation du module).

De plus, les programmes devront avoir une exécution entièrement automatique, puisque l'ensemble du système doit se faire sans intervention humaine dans la mesure du possible.



Cette méthodologie a pour avantage de n'imposer, par exemple, aucune contrainte de développement au niveau du choix du langage de programmation à la communauté et de standardiser un minimum les contributions au consortium. Quant au choix des données en entrée et en sortie des différents plugins, il pourrait être comme l'illustre ce schéma :

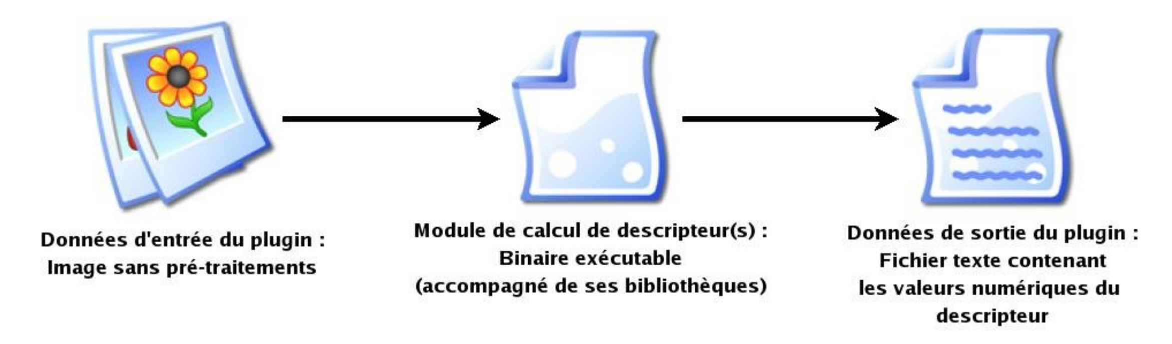


FIG. 4.8 – Standard de développement de plugins (Calcul de descripteur)

Troisième partie

Développement de la plateforme Intra/Internet

Chapitre 5

Méthodologie et travail réalisé

5.1 La gestion de projet

Ce stage était un projet informatique « comme un autre ». On peut donc supposer que l'utilisation d'une méthodologie de gestion de projet informatique aurait été logique. Cependant, l'utilisation d'une méthode « stricte » était en fait peu adaptée à ce cas pour différents motifs :

- Dans le cadre de projets menés par une seule personne (comme c'était le cas ici), la mise en place de mécanismes complets de management de projet peut s'avérer complexe et nécessiter plus de temps qu'elle n'en fait gagner.
  Nous avons donc adopté une gestion de projet minimale, essentiellement basée sur les comptes-rendus, en moyenne hebdomadaires, lors d'entretiens individuels ou de petites réunions.
- Enfin, comme l'approche employée est de type empirique, son évolution est par essence difficile à prévoir (car on traite les problèmes au fur et à mesure de leur découverte). Une prévision trop précise et un planning décidé trop tôt auraient été remis en cause à chaque nouveau problème.

5.1.1 Communication au sein du consortium

Le projet étant au centre d'une vaste communauté scientifique, il m'avait été demandé de réanimer les échanges au sein du consortium dans le but d'effectuer, par la suite, une synthèse afin de visualiser rapidement les travaux de chaque laboratoire. Certains aspects de communication ont pu être abordés au sein d'échanges électroniques ou même au cours d'un séminaire effectué à la Bresse. Cependant, il s'est vite avéré que cet aspect serait très difficile à mettre en avant étant donné les problèmes de communication persistant au sein du groupe.

5.2 La démarche suivie

Après s'être replongé dans le contexte de Madonne, il a d'abord été nécessaire de s'imprégner et de synthétiser tout ce qui avait été effectué lors de l'unité d'enseignement Vision 2 [1] :

- lire et comprendre les documents rédigés par les étudiants.
- trier les idées et autres réflexions faites au cours de cette U.E.

[1] Unité d'enseignement suivie au premier semestre du Master 2 IMA et dirigée par Jean-Marc OGIER



- répertorier les outils, documentations et autres pointeurs utilisés par les étudiants.

Une fois ce travail préalable effectué, l'étude des problématiques (cf. Chapitre 3, page 11) a donc été complétée ou revue puis le projet a donc été découpé, un peu à la manière de Vision 2 en quatre grands « domaines » qui seront développés par la suite :

- Développement de l'interface utilisateur (cf. Partie 5.4, page 40) ;
- Développement et interfaçage de modules de calcul de descripteurs(cf. Partie 5.5, page 46) ;
- Développement de la base de données (cf. Partie 5.7, page 68).
- Développement de la couche réseau et sécurité (cf. Partie 5.8, page 72) ;

Néanmoins, il s'est rapidement avéré qu'une cinquième partie, qui devait être seulement anodine, représente une charge de travail considérable :

- Administration système (Installation, configuration, déploiement, test et maintenance) (cf. Annexe B, page 98)

5.3 Préparation de l'environnement de travail

Pour pouvoir développer correctement l'application, il était nécessaire de posséder des droits différents de ceux de simples utilisateurs. En effet, pour pouvoir travailler dans de bonnes conditions, je devais pouvoir :

- Installer des applications, bibliothèques et autres outils
- Accès à la configuration des systèmes sur lesquels l'application interviendrait
- Installer et configurer des démons
- etc...

Pour des raisons de sécurité et de logique, il était impensable d'effectuer de telles actions sur un environnement de production, ni de posséder de tels droits sur le réseau du laboratoire. Je devais donc créer un environnement similaire dit « de développement » sur une machine locale coupée du réseau.

Une solution envisageable était d'effectuer la demande d'un ordinateur qui serait en permanence disponible, totalement coupé du réseau de recherche et où les droits d'administrateur m'étaient octroyés. Néanmoins, aucun poste fixe ne répondait à tous ces critères.

Ne voulant pas perdre trop de temps à effectuer des démarches administratives qui ralentiraient l'avancée de mon stage et possédant personnellement un ordinateur portable sous environnement Linux Fedora Core 3, sur lequel des droits d'administrateur me sont octroyés, je me suis décidé à l'utiliser dans le cadre du projet.

Conformément à l'étude qui a été faite précédemment (cf. Sous-partie 4.3.1, page 28) sur les différentes solutions web, j'ai donc installé et configuré la plate-forme LAMP (Manuel d'installation : cf. Annexe B.3, page 101) et ai commencé le développement de quelques pages en PHP en m'inspirant de certaines maquettes effectuées dans le cadre de l'UE Vision 2.



5.4 Développement de l'interface Utilisateur

Dans un souci de clarté et afin que mon travail soit le plus compréhensible possible et modifiable par la suite, j'ai décidé de m'imposer quelques contraintes en terme de développement PHP :

1. Développer en suivant une méthodologie de programmation modulaire et objet même si, à l'origine, PHP est surtout un langage procédural (la couche objet du langage n'est apparue que depuis la version 4 et est loin d'être complète, de même pour la dernière version).
2. Séparer autant que possible la partie purement graphique du reste de l'application.
3. Utiliser des noms de variables, de fonctions, etc... les plus clairs possibles tout en veillant à un anglicisme maximum (afin que tout développeur, quelle que soit sa nation, puisse comprendre le code source).
4. N'utiliser aucune constante au sein même de fonctions ou de classes et privilégier systématiquement la création de variables globales quand cela est possible pour une modification et/ou configuration plus aisée.
5. Documenter un maximum l'intégralité de mon code source.

Chaque cas d'utilisation possède son propre fichier PHP (insérer des images, des plugins et effectuer une recherche) et différentes classes ont été créées symbolisant les objets « vivants » qui sont manipulés par l'application (images et ses informations, arbres XML, différents types de connections, etc...).

En ce qui concerne la partie purement graphique de l'application, elle a été dissociée du code PHP pur, tant que cela était possible, par l'utilisation d'une feuille de styles CSS [2] (aussi rudimentaire soit elle) et de templates qui pourront ainsi facilement être modifiables par un graphiste dans le but d'améliorer l'ergonomie de la plate-forme.

5.4.1 Utilisation des templates

Un template est un anglicisme utilisé en informatique pour désigner un modèle de conception de logiciel ou de présentation des données. On parle aussi de «patron» comme en couture.

Un template est un moyen de séparer le fond (le contenu informationnel) de la forme (la manière dont il est présenté).

Très utilisé dans la conception de sites web, un template agit comme un modèle dans lequel seuls certains éléments sont modifiables (le contenu). Cela facilite la conception et la mise à jour des sites, aussi bien sur le contenu, que sur la présentation.

- changer la charte graphique du site revient à changer le template et cela met à jour toutes les pages du site ;
- ajouter une page ne consiste plus qu'à en écrire le contenu.

[2]Cascading Styles Sheets



Les templates dans le code de la plate-forme

En ce qui concerne l'implémentation des templates, la classe *Template* de la bibliothèque *PHPLib* [3] a été utilisée. Cette classe peut être extraite de l'ensemble de la bibliothèque puisqu'elle est codée dans un simple fichier : *templates.inc*. Ainsi il suffit de l'inclure dans chaque script où l'on désire utiliser les templates via la commande : *require_once(path/templates.inc)* ou une commande similaire.

Lors de l'utilisation des templates, il faut donc construire des modèles - fichiers TPL - qui serviront à la génération des fichiers HTML affichés dans le navigateur. Schématiquement, ces fichiers contiennent des balises HTML traditionnelles ainsi que des balises spéciales entre accolades ou sous forme de commentaires HTML spéciaux (<!– ->). En fait, ces balises font office de variables dont les valeurs seront précisées dans les scripts PHP.

Le fonctionnement est des plus simple :

1. On instancie un objet de la classe Template en lui passant comme paramètre le chemin du répertoire contenant les modèles
2. On précise ensuite via la méthode *set_file* le fichier du modèle que l'on va utiliser
3. On affecte à chaque variable utile du modèle une valeur via la méthode *set_var*
4. On lance l'analyse permettant de remplacer les balises par leurs valeurs dans le futur code HTML, par le biais de la méthode *parse*
5. Une fois tout ceci terminé, on lance l'affichage sur le navigateur par la méthode *print* (cette étape ainsi que la précédente peuvent être effectuées en même temps grâce à la méthode *pparse*).

Un des gros intérêts des templates est également de pouvoir factoriser du code de présentation qui se répète en le symbolisant par un bloc dont le contenu sera dynamique. On devra au préalable déclarer au sein du modèle le nom du bloc ainsi que son emplacement de la manière suivante :

```
...
<!-- BEGIN MON_BLOC -->
...
{ma_variable}
...
<!-- END MON_BLOC -->
...
```

L'utilisation de ce bloc pourra alors se faire au sein d'une boucle où le nombre d'itérations définira le nombre de fois où le bloc sera répété. On peut donc affecter alors à la variable *ma_variable* autant de valeurs que l'on veut et ce dynamiquement au sein, par exemple, d'une boucle *for* ou *while*. Le code PHP permettant d'utiliser le bloc précédent est le suivant :

[3] http://phplib.sourceforge.net/



```
// Déclaration du bloc MON_BLOC
$template->set_block("MonModèleTPL","MON_BLOC","mon_bloc");
foreach($i=0;$i<10;$i++)
{
  $template->set_var("ma_variable",$i);
  $template->parse("mon_bloc","MON_BLOC",true);
}
```

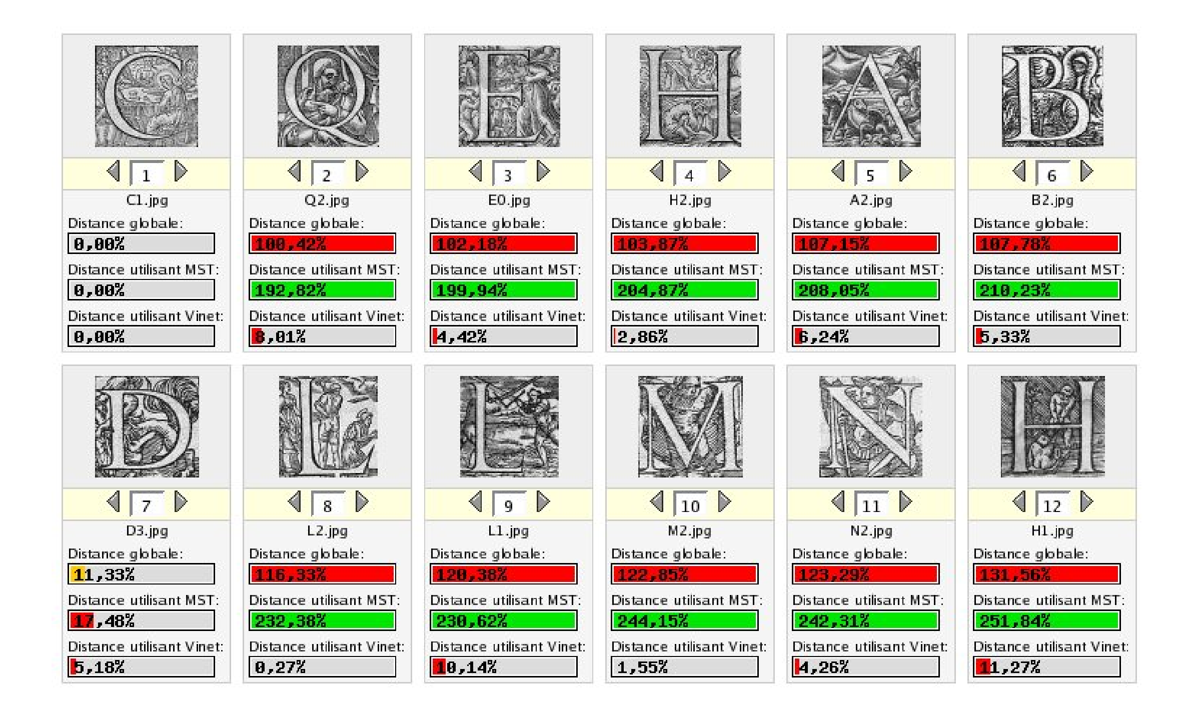


FIG. 5.1 – Exemple d'affichage résultant d'une requête de recherche d'images

L'utilisation de templates dans notre projet devient particulièrement intéressant lorsqu'on retrouve à plusieurs reprises le même type de représentation graphique. En effet, lors de l'affichage des résultats d'une requête de recherche d'images similaires (cf. figure 5.1, page 42), on obtient une sorte de tableau contenant des images avec un certain nombre d'informations qui se répète (Affichage des valeurs des **n** descripteurs pour chaque image). On utilisera donc deux blocs imbriqués : un pour l'affichage des **m** images du résultat et l'autre pour les **n** descripteurs de chaque image.

5.4.2 Configuration de la plate-forme

L'utilisation de variables globales d'environnement permet le regroupement des constantes dans des emplacements bien précis (ex. : les fichiers env_vars.php et *config.php*) afin de rendre l'application moins figée et plus paramétrable (modifier l'emplacement des fichiers, les types de fichiers acceptés, les adresses IP des serveurs utilisés, etc. . .).

Grâce à ce procédé, la plate-forme est entièrement paramètrable et modulable per-



mettant de fonctionner tout aussi bien sur un ou plusieurs machines avec une répartition hétérogène de ses fichiers.

Voici ce qu'on peut rencontrer dans le fichier *config.php* :

```
...
// Configuration de l'annuaire LDAP
$LDAP_SERVER[0]="indus.univ-lr.fr";
//Port utilisé par défaut par le serveur LDAP
$LDAP_PORT[0]="389";
$LDAP_ROOT[0]="dc=my-domain,dc=com";
$LDAP_ROOT_DN[0]="cn=Manager,dc=my-domain,dc=com";
$LDAP_ROOT_PW[0]="secret";

// Configuration du serveur MySQL
$mysql_server="indus.univ-lr.fr";
$mysql_user="root";
$mysql_password="jmogier";
$mysql_db="madonne_web";
...
```

Et les variables intéressantes du fichier *env_vars.php* :

```
//chemin absolu par rapport à la racine du serveur web
$website_absolutepath="/madonne_web/";
$tmp_dir_default="/tmp/";
$cache_dir_default=$website."cache/";
$plugins_dir_default=$website."cgi-bin/";
$templates_dir_default=$website."templates/";
$classes_dir_default=$website."classes/";
$functions_dir_default=$website."functions/";
$images_dir_default=$website."images/";
$thumbs_dir_default=$images_dir_default."thumbs/";
$xml_dir_default=$images_dir_default."xml/";
$image_convert_cmd="convert ";
$remove_files_cmd="rm -rf ";
$signatures_table="signatures";
$images_table="images";
$DEBUG=false;
$xmlrpc_mode=true;
$xmlrpc_debug_mode=false;
$xmlrpc_serverfilepath="xmlrpc_server.php";
$compute_server_host="moca.univ-lr.fr";
error_reporting(E_ALL & ~E_NOTICE);
// Taille maximale autorisée pour l'upload de fichiers
$file_max_size_default=2097152;
// Tableau des types mimes autorisés pour les images
$mime_types_default=array('image/jpeg',
                          'image/pjpeg',
```



```
                              'image/gif',
                              'image/x-ms-bmp',
                              'image/tiff');
$nb_images_default=10;
$decimal_precision_default=2;
...
```

Le nombre de possibilités de paramétrage est donc assez importante ce qui en fait un véritable atout pour la plate-forme.

5.4.3 Documentation

En ce qui concerne la documentation du code, chaque fichier a été commenté selon le standard utilisé dans DOxygen afin que ce dernier puisse générer plusieurs types de documentation utiles pour le développeur.

5.4.4 Structure et organisation

Outre le développement orienté objet, l'ensemble des scripts écrits respecte l'organisation suivante :

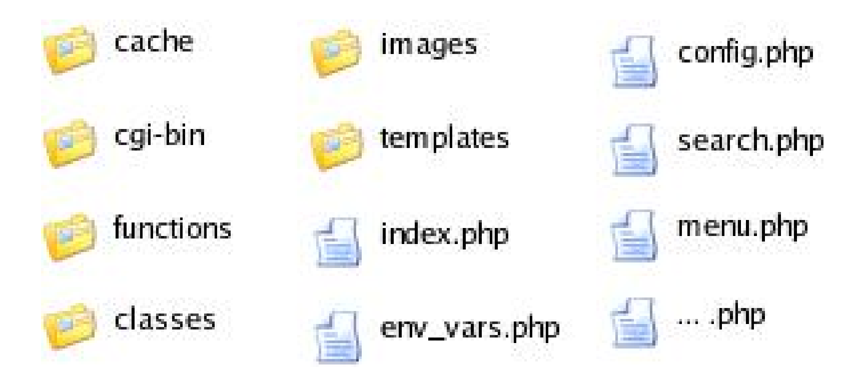


FIG. 5.2 – Arborescence de l'application

- Le dossier *cache* contient toutes les données temporaires de la plate-forme qui seront supprimées à la fin de la session lorsque l'utilisateur se déconnectera de celle-ci.
- Le dossier *cgi-bin* contient tous les binaires des plugins qui ont été insérés dans la plate-forme.
- Le dossier *classes* contient toutes les définitions de classes PHP utilisées dans la plate-forme.
- Le dossier *functions* contient tous les fichiers regroupant des fonctions non-incluses dans les classes PHP.
- Le dossier *images* contient l'ensemble des images uploadées dans le format JPG ainsi qu'une version miniaturisée et les fichiers XML associés contenus dans le sous-répertoire *xml*.



- Le dossier *templates* contient l'ensemble des modèles utilisés concernant l'affichage purement graphique de l'application.
- Le fichier *index.php* représente l'accueil de l'application.
- Le fichier *config.php* contient l'ensemble des données relatives aux identifiants et mot de passes des bases de données et serveurs annexes utilisés par l'application.
- Le fichier *env_vars.php* contient l'ensemble des variables de configuration de l'application.
- Le fichier *menu.php* représente le menu affiché une fois authentifié sur l'application.
- Le fichier *search.php* représente la page de recherche de similarités entre images.
- etc...

Ces critères de développement, qui peuvent ainsi permettre de gagner un temps précieux pour la reprise et/ou la compréhension de mes travaux, semblent simples et implicites mais s'ils n'avaient pas été respectés, cela aurait rendu très laborieux la réutilisation et l'amélioration de l'application par une personne tierce.

5.5 Traitement d'images

Afin de manipuler les images de la base, il a fallu effectuer un choix quant à la bibliothèque (cf. Partie 4.1, page 21) à utiliser. Celle-ci devait permettre le chargement et l'enregistrement dans un nombre assez conséquent de formats intéressants et offrir des fonctionnalités avancées quant au traitement d'images. Après quelques discussions au sein du laboratoire, il s'est avéré qu'un grand nombre de chercheurs effectuait déjà leurs traitements en utilisant la bibliothèque OpenCV [4] d'Intel. Outre le fait que cette dernière est capable de lire des formats standards comme le JPEG, le PNG, le BMP et le TIFF, elle possède surtout une API très riche en terme d'imagerie. De surcroît, cette dernière étant multi-plate-forme, elle s'est très vite imposée comme étant un très bon choix.

5.5.1 Présentation de la Bibliothèque OpenCV d'Intel

cf. Annexe C, page 109

5.5.2 Développement de plugins OpenCV

Les traitements devant s'interfacer facilement avec la plate-forme (cf. Sous-partie 3.3.4, page 19), les plugins ont du être développés avec la possibilité de s'exécuter en ligne de commande sans interface graphique. Ceux-ci ont également pour but de limiter le nombre d'interventions humaines dans les calculs et ainsi éviter des coupures dans le fil global d'exécution (cf. Sous-partie 3.3.3, page 19).

Nous obtenons alors des modules de calculs totalement autonomes (hormis leur dépendance avec OpenCV) qui n'ont besoin que de données d'entrée pour produire un résultat qui sera ensuite interprété par la plate-forme. Ces modules ou plugins peuvent donc être réutilisés au sein d'autres systèmes et/ou intégrés facilement dans des chaînes de traitements de type *Batch*.

Création d'un plugin de test

Dans le but d'obtenir rapidement une première version de démonstration des fonctionnalités de la plate-forme, un premier module permettant de calculer une signature simple a été créé. Cette signature consiste simplement à effectuer une distance de Vinet (Somme des distances entre pixels de deux images) entre une image de référence et une autre image unie de couleur blanche et de mêmes dimensions. Ce type de signature n'est pas très pertinent puisqu'il n'est que peu représentatif de l'image. Il se contente d'illustrer la proportion colorimétrique de l'image dans sa globalité et est donc peu robuste aux dégradations de celle-ci.

Le développement de ce premier module devait également permettre de mettre en oeuvre une première chaîne complète d'utilisation de la plate-forme afin de proposer une première approche de l'interopérabilité (cf. Sous-partie 3.3.4, page 19) du système global. Le module devait donc utiliser une simple image en entrée et produire comme

[4] Bibliothèque de traitement d'images écrite en C++ par Intel

résultat un ensemble de valeurs numériques (ici un nombre unique représentant la distance globale de l'image par rapport à une image unie de couleur blanche et de mêmes dimensions).

Ce résultat devant être transmis à la plate-forme, et afin de ne pas compliquer les processus de communication, il a été décidé d'utiliser un fichier temporaire dans lequel serait écrit le résultat de l'exécution du plugin. Ce fichier serait ensuite lu et interprété par la plate-forme avant d'être effacé. Dans un premier temps, nous avons utilisé un simple fichier au format ASCII avec une organisation tabulaire, bien qu'une organisation semblable au XML aurait pu être un plus, s'il avait été couplé d'un parseur adéquat ; mais cela aurait ralenti l'avancé du développement.

Pour faciliter et normaliser la communication entre plate-forme et plugins, chaque plugin devra suivre un ensemble de règles bien définies :

1. le nom du binaire exécutable principal devra être une forme condensée du ou des descripteurs qu'il représente, ceci afin de faciliter son appel et son identification.
2. l'entrée du plugin se limitera à un chemin représenté par une chaîne de caractères vers l'image sur laquelle il pourra s'appliquer.
3. aucune intervention humaine n'est possible entre le début et la fin de son exécution afin de rendre l'ensemble du traitement non supervisé.
4. l'exécution devra pouvoir s'effectuer entièrement sans interface graphique.
5. le plugin devra pouvoir s'exécuter et/ou se compiler sur n'importe quelle machine et être livré avec toutes ses dépendances en terme de bibliothèques externes (exemple OpenCV).
6. le résultat du plugin devra être un fichier texte contenant l'ensemble des valeurs numériques relatives au(x) descripteur(s) qu'il représente. Dans un premier temps, celui-ci devra d'organiser de la façon suivante : une ligne → un nombre.
7. le nom du fichier résultat devra être codé de la façon suivante : *nomdufichierimage***.***nomduplugin*. De plus, l'emplacement du fichier généré devra être paramétrable et sera par défaut l'emplacement système des fichiers temporaires (ex. : */tmp/* sous Unix) afin que le fichier soit facilement repérable par la plate-forme.
8. le calcul ne devra pas non plus demander trop de temps d'exécution afin de ne pas pénaliser le fonctionnement de l'intégralité de la plate-forme.

Le premier plugin nommé *Vinet* a donc servi de modèle de développement pour les plugins suivants. Les sources de ce dernier s'organise alors de la façon suivante :

– les fichiers *Manager.cpp* et *Manager.h* relatifs à la déclaration et la définition d'une classe C++ générique définissant les mécanismes de chargement et de sauvegarde de fichiers (interface avec OpenCV).
– le fichier *Signature.h* définissant le prototype de la classe générique *Signature*.
– les fichiers *Vinet_Utils.cpp* et *Vinet_Utils.h* contenant uniquement ce qui est propre aux calculs des descripteurs.
– le fichier *Vinet.cpp* étant le lien entre ces différents éléments contiendra le *main* et sera donc le fichier principal.

– un fichier *Makefile* comportant l'ensemble des directives de compilation et d'édition de liens avec le compilateur (par ex. : GCC) permettant ainsi une compilation simple et multi-plate-forme puisque GCC et Make sont dorénavant existants sur plusieurs environnements systèmes.

Le développement des autres plugins pourra donc s'inspirer de ce type de structure quoique non obligatoire si les plugins respectent les règles citées plus haut.

Après avoir testé le premier plugin de manière isolée et de façon manuelle sur différentes images et supposé de son bon fonctionnement au sein de la plate-forme, nous avons décidé de poursuivre le développement d'autres plugins afin d'élargir les futurs tests.

Migration de traitements déjà existants

Suite à nos entretiens réguliers et aux discussions menées durant les réunions, il s'est avéré intéressant d'intégrer au sein de plugins, les différents traitements disponibles dans la plate-forme Windows de Mario Di Carlo. En effet, son travail s'insérait dans le domaine de la reconnaissance de symboles. Les traitements utilisés dans la plate-forme qu'il avait développée nécessitait des calculs de descripteurs sur les images, calculs présentant un fort intérêt pour la plate-forme Madonne. Parmi ces calculs, trois semblaient vraiment intéressants :

– la transformée de Fourier-Mellin
– les polynômes de Zernike
– les sondes circulaires

La transformée de Fourier-Mellin était un outil développé dans le cadre de la thèse de S. Adam. Les moments invariants issus de son calcul permettent de caractériser l'image quelles que soient sa rotation et sa taille (moyennant l'utilisation du théorème du retard que l'on retrouve dans l'analyse de Fourier ainsi qu'une normalisation).

Les moments invariants de Zernike issus du calcul des polynômes de Zernike, constituent le modèle le plus fréquemment utilisé dans la littérature de la reconnaissance (de nombreux auteurs se sont inspirés de cette approche en utilisant les travaux de M. Teague) de formes invariantes aux rotations et aux changements d'échelle.



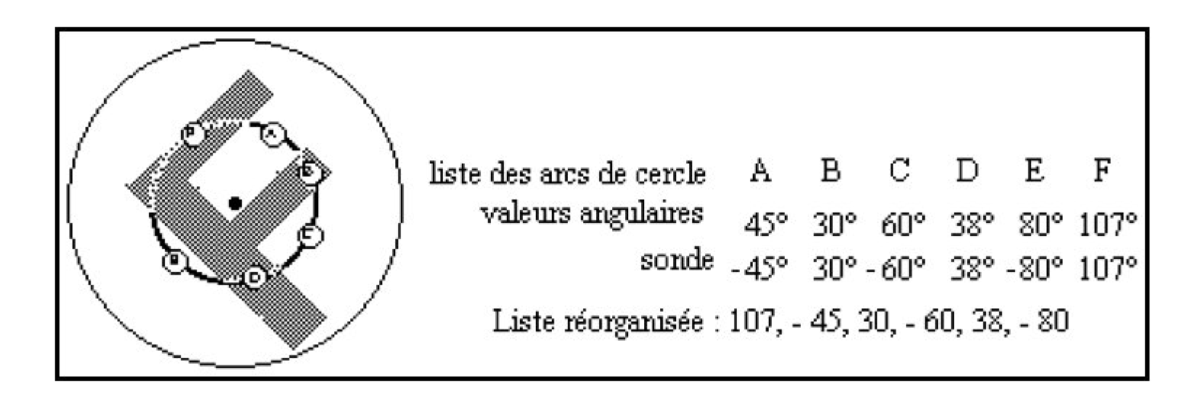


FIG. 5.3 – Méthode de reconnaissance de formes par sondes circulaires

La méthode utilisant les sondes circulaires, développée dans le cadre de la thèse de L. Lefrère, consiste à utiliser des valeurs angulaires obtenues à partir de cercles concentriques dont le centre est le barycentre de la forme à caractériser. On obtient cette liste en parcourant les cercles dans un sens donné et en enregistrant lors du parcours, les écarts angulaires existants entre les points d'entrée et de sortie de la forme.

Étude rapide de l'existant

A la lecture du rapport de M. Di Carlo, il s'est avéré que la partie interface utilisateur et l'aspect traitement avaient déjà été scindés en deux groupes. Il suffisait donc en théorie de créer une interface légère de type *ligne de commande* pour pouvoir rendre chaque traitement indépendant. Cependant, après avoir perdu beaucoup de temps à rendre le code compilable sous Linux avec *GCC* (erreurs de compilation nombreuses dues à la structure un peu désordonnée des fichiers sources et des standards de programmation non respectés peut-être dus au développement effectué sous Microsoft Visual Studio), nous nous sommes rendus compte d'une première difficulté : Le caractère restrictif concernant les formats d'entrée et de sortie des fichiers images.

En effet, les seuls formats d'entrée acceptés sont le BMP, le PGM, le PCX et en sortie le BMP, PBM et PPM ; ce qui ne correspondait pas à nos attentes. Il a donc été décidé de réécrire une partie du code afin d'utiliser la bibliothèque OpenCV qui propose plus de possibilités de formats d'importation et d'exportation.

Migration

Le programme de Di Carlo étant basé sur la classe centrale *CImage*, il suffisait donc de la réécrire complètement (et de la renommer afin d'éviter un conflit avec une classe existante du même nom dans OpenCV) en développant un wrapper basé sur la structure de données *IplImage* de OpenCV qui contient toutes les informations de l'image. Bien qu'ayant réécrit une grande partie du code de cette classe, le résultat n'a pas été un succès puisque le code source, bien que ce compilant correctement, continue de s'exécuter anormalement (résultats des traitements incohérents, fuite de mémoire,



etc...). Le choix d'abandonner la migration a donc été fait pour ne pas perdre inutilement de temps sur le débogage de la bibliothèque de Di Carlo.

Intégration d'un plugin basé sur le Minimum Spanning Tree

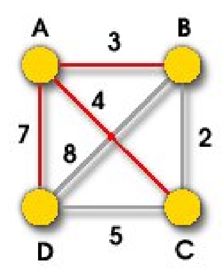
Dans le même contexte que celui du développement la plate-forme Madonne, une partie du travail de Mouhammed Hammoud a consisté à développer un programme permettant de calculer le MST [5] sur une image ayant subi en amont un pré-traitement. Mon rôle étant ensuite de pouvoir intégrer ce calcul au sein d'un plugin utilisable dans la plate-forme.

L'intérêt d'utiliser le MST comme descripteur est qu'il permet de décrire l'organisation globale des éléments composant l'image. L'organisation d'une image est de ce fait invariante à la rotation et à la translation. Et il suffit d'effectuer une normalisation pour qu'elle soit invariante au changement d'échelle.

Bref rappel concernant le MST

Le problème de l'arbre couvrant de poids minimum (ou MST, Minimum Spanning Tree) consiste à trouver un arbre couvrant dont la somme des poids des arêtes est minimum.

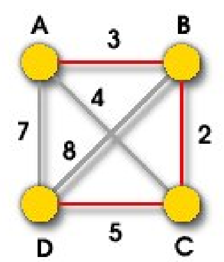
La seule condition, nécessaire et suffisante, pour qu'un graphe admette un arbre couvrant est qu'il soit connexe. Un arbre couvrant de poids minimum est en général différent de l'arbre des plus courts chemins (SPT, Shortest Paths Tree) construit par exemple par BFS ou Dijsktra. Un arbre des plus courts chemins est bien un arbre couvrant, mais il minimise la distance de la racine à chaque sommet, et non la somme des poids des arêtes.



Un arbre des plus courts chemins (SPT) de racine **A**

Sa hauteur est **7**

Son poids est **14**



Un arbre de poids minimum (MST)

Sa hauteur est **10**

Son poids est **10**

FIG. 5.4 – Le Minimum Spanning Tree (MST) par rapport au Shortest Path Tree (SPT)

Il existe deux algorithmes célèbres pour résoudre le problème de l'arbre couvrant

[5] Minimum Spanning Tree



de poids minimum. Chacun d'eux utilise plus particulièrement l'une des caractérisations des arbres pour trouver un MST : soit en considérant les arbres comme des graphes connexes avec le minimum d'arêtes, soit en considérant les arbres comme des graphes acycliques avec le maximum d'arêtes. Ces algorithmes utilisent également deux techniques de résolution très différentes :

– Algorithme de Prim : Il maintient au fur et à mesure de la construction un sous-graphe connexe qui grossit petit à petit.
– Algorithme de Kruskal : Il maintient au fur et à mesure de la construction un graphe partiel acyclique. Si les algorithmes de recherche sont spécifiques aux graphes, l'algorithme de Kruskal utilise lui un paradigme de résolution plus général : les algorithmes gloutons.

Présentation de la chaîne complète du plugin

Il a été précisé précédemment dans ce rapport que tout plugin doit avoir en entrée une image n'ayant subi aucun pré-traitement. Or, le calcul du MST d'une image s'effectue sur les régions d'une image, noeud de l'arbre. En réalité, pour fonctionner, le programme de Mouhammed a besoin que l'image soit segmentée en un ensemble de couches distinctes (zones homogènes de l'image, zones texturées, etc...) où chaque couche est représentée par un ensemble de régions (ayant elles-mêmes subi des post-traitements) unies de niveaux de gris différents afin de les identifier. Le travail de segmentation est effectué via un programme développé sous Matlab pour Surapong Uttama dans le cadre de ses recherches. Le plug-in devrait donc en théorie regrouper l'ensemble des calculs et se schématiser de la façon suivante :

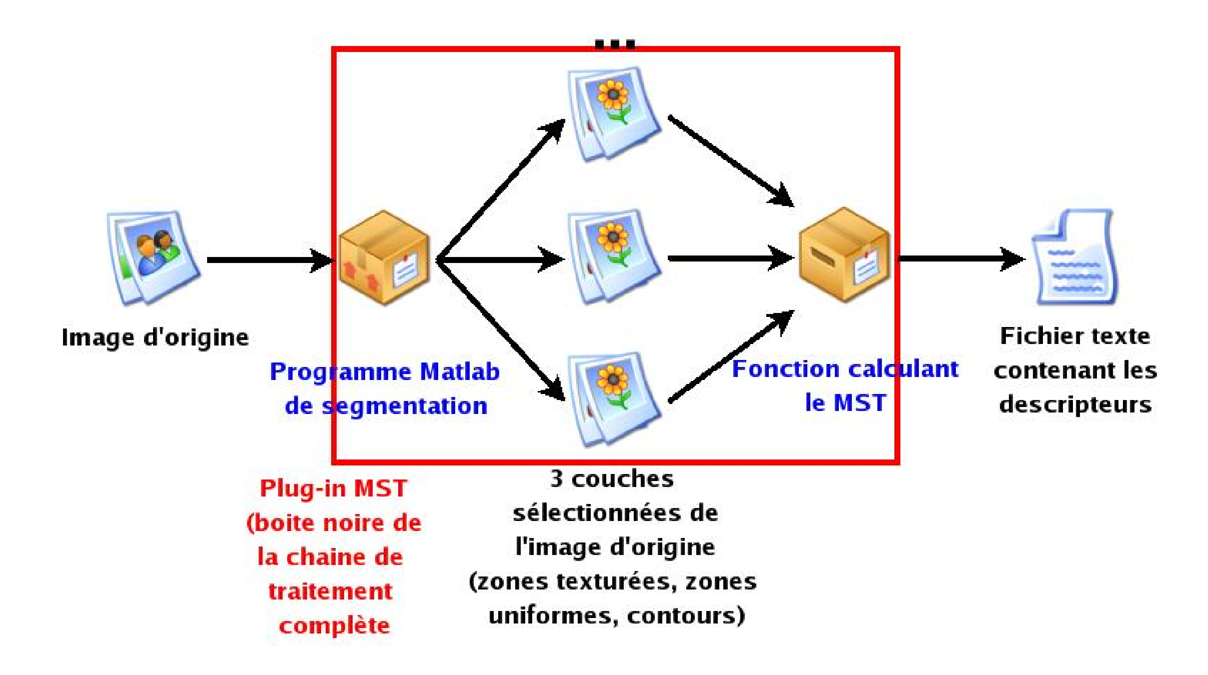


FIG. 5.5 – Chaîne complète du calcul effectué par le plugin MST



Bien que le calcul des MST de chaque couche soit entièrement non-supervisé, le pré-traitement en amont pour la segmentation pose problème. En effet, le programme actuel (développé sous Matlab) contient bon nombre de paramètres à fixer humainement lors de l'exécution de ce dernier. Même si un certain nombre d'entre eux puisse être un jour fixé avec des valeurs par défaut, d'autres en revanche posent problème de par leur subjectivité. Par exemple, le nombre de couches générées par la segmentation, est actuellement une valeur fixée à la main en fonction de la complexité de l'image qui reste un critère fort subjectif. Enfin, le programme Matlab génère les couches segmentées de façon non ordonnée. Bien que l'être humain puisse les distinguer et donne un sens à chaque couche, le programme, à l'heure actuelle, en est bien incapable. Ceci pose de gros soucis lorsqu'on désire comparer les MSTs de différentes images puisque pour que la comparaison ait un sens, il est nécessaire de comparer des valeurs de MST de couche de même nature ; or ces dernières ne sont pas ordonnées automatiquement mais manuellement.

Cependant, afin de pouvoir quand même tester la pertinence d'un descripteur comme le MST au sein de la plate-forme Madonne, certaines astuces concernant l'intégration propre à cette chaîne de traitements ont été utilisées.

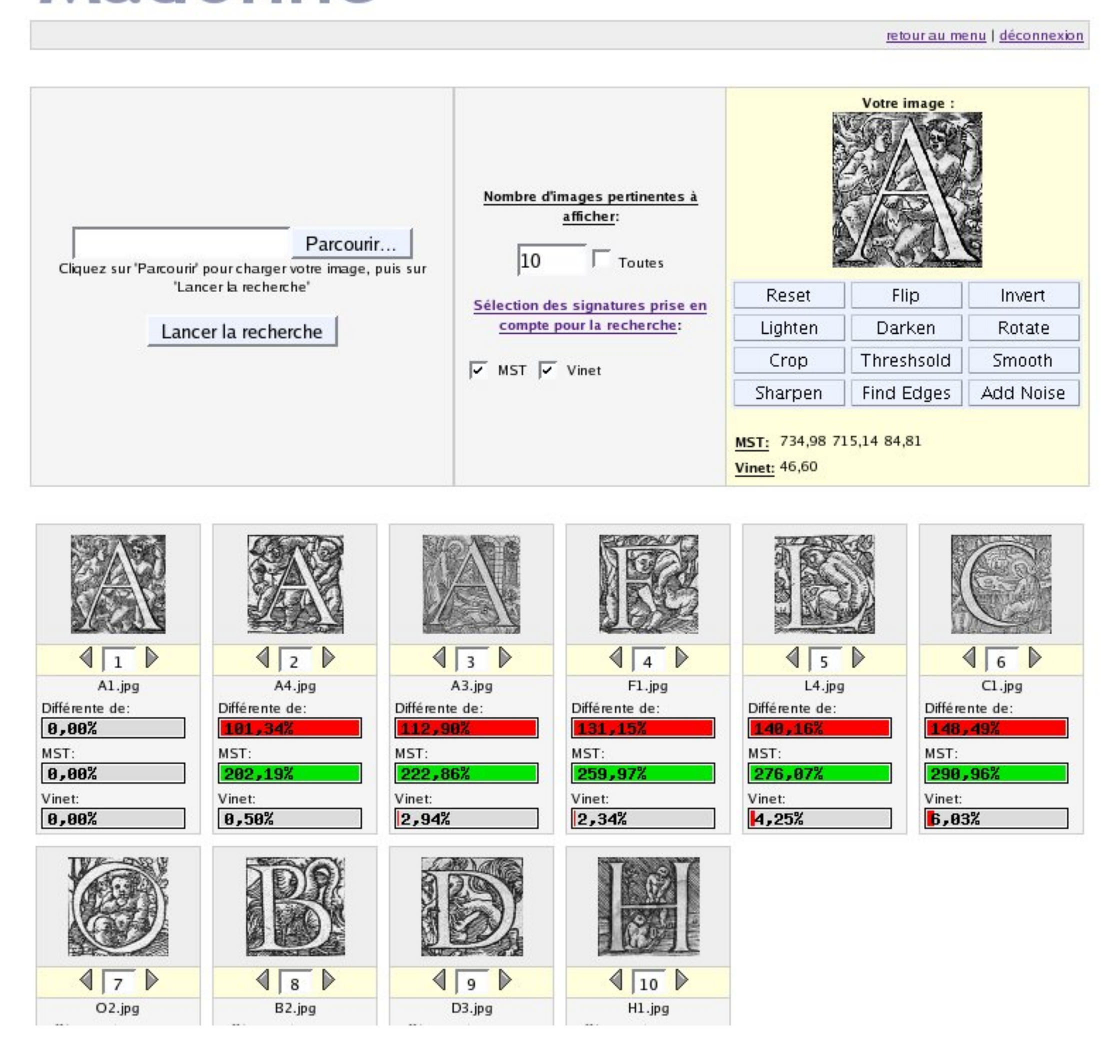


FIG. 5.6 – Exemple de résultats obtenus lors de la recherche de similarité d'une image au sein de la base

Jeu de données utilisés

Le consortium Madonne s'inscrivant dans un contexte de documents anciens, et la plate-forme devant travailler sur des images, les données mises à notre disposition furent des lettrines en niveau de gris et fortement illustrées. La segmentation étant très sensible à la dégradation de la qualité de numérisation, seules les images au format Bitmap et de bonne qualité ont été retenues parmi une base d'origine importante (CESR [6]).

Premièrement, nous avons fixé le nombre de couches, utilisées dans la recherche de

[6]Centre d'études supérieures de la renaissance de Tours



similarités entre images, à 3. Ensuite, nous avons demandé à S. Uttama de nous préparer un jeu de test sur lequel des recherches pourront être effectuées. Cela représentant un travail important pour S. Uttama en terme de temps de travail, nous n'avons pu recueillir qu'une petite trentaine d'images segmentées (30 x 3 couches tout de même). Chaque image ayant subi par la suite la normalisation suivante :

- L'image d'origine sera nommée en fonction de la lettre de la lettrine suivie d'un numéro pour distinguer les lettrines de même lettre (ex. : A1, A2, G5, etc...)
- Chaque couche sera nommée en utilisant, comme son image d'origine, la lettre de la lettrine suivie du numéro plus le suffixe *_label* suivie encore d'un numéro différenciant les trois couches.
- chaque type de couche portera toujours le même numéro relatif à sa nature.
- l'ensemble des couches a été placé dans un même répertoire nommé *layers*

Les images intermédiaires ont été conservées à cause de leur intérêt dans d'éventuels autres traitements.

Grâce à ce schéma, il est alors aisé d'associer à chaque image de la base de test chacune de ses couches de par son nom et son emplacement connu. Il suffit donc, en suivant l'architecture de développement de plug-ins suggérée plus haut, d'indiquer, dans le programme principal *MST.cpp*, la façon de récupérer les couches à partir de l'image d'origine spécifiée en entrée du plug-in.

Source 5.1 – Astuce utilisée pour le chargement des couches du MST

```
...
#define NB_LAYERS 3
#define LAYERS_PATH "/var/www/html/madonne_web/layers/"
...
int main(int argc, char* argv[])
{
  IplImage* img[NB_LAYERS];
  MST* mst;
  /* Explication sur les différents paramètres:
  * - argv[0] correspond au nom du programme d'exécution
  * - argv[1] correspond au chemin de l'image
  * - argv[2] correspond au chemin absolue dans
  * lequel sera stocké le fichier contenant la signature
  * (optionnel, par défaut il vaut /tmp sous Unix
  */
  switch(argc)
  {
    case 2:
      for(int i=0;i<NB_LAYERS;i++)
      {
        string old_filename=(string)basename(argv[1]);
        ...
        string new_extension="_label"+num+".bmp";
        old_filename.replace(old_filename.length()-4,4,new_extension
            );

        string filepath=LAYERS_PATH+(string)old_filename;
        ...
        readImage((char *)(filepath.c_str()),&img[i],false);
```



```
        }
        mst=new MST(img,(string)basename(argv[0]));
        mst->savetoFile((string)argv[1]);
        ...
    }
    ...
}
```

Comme l'illustre cet extrait du code source du fichier *MST.cpp*, on construit un tableau de 3 *IplImage* représentant chaque couche à partir du nom du fichier d'origine. On fournit ensuite ce tableau à la fonction de calcul de MST. L'architecture d'origine est donc conservée, à savoir : une image sans pré- traitement en entrée, un fichier texte en sortie. Cette astuce a bien évidemment quelques inconvénients mais reste à priori un bon compromis en attendant une chaîne de traitement moins supervisée.

5.5.3 Interactions entre plugins et images

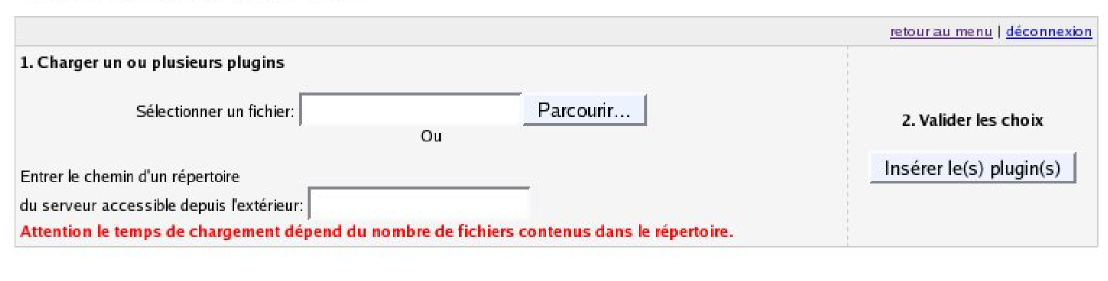


FIG. 5.7 – Page relative à l'insertion de plugins

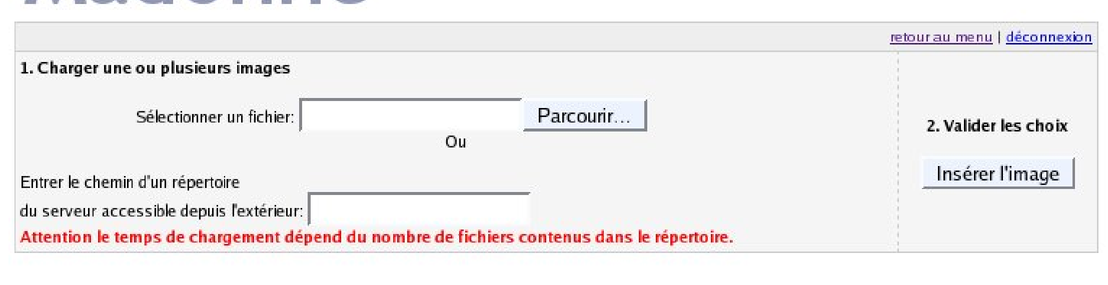


FIG. 5.8 – Page relative à l'insertion d'images



Lorsqu'une image ou un plugin est inséré, il faut pouvoir mémoriser son emplacement afin que l'application ait en sa possession une liste des ressources qui lui sont disponibles. Ceci est donc effectué lors de ces insertions par la mémorisation des noms, chemins et autres informations (« poids » d'un plugin, nombre de paramètres, etc...) dans des tables d'une base de données relationnelle en l'occurrence MySQL.

Cependant, l'insertion de plugins et d'images dans la plate-forme soulève un certain nombre d'autres questions auxquelles il a fallu apporter des solutions. Par exemple :

– Que doit-il se passer au moment de l'insertion de nouveaux plugins ?
– Que doit-il se passer au moment de l'insertion de nouvelles images ?

En effet, des interfaces bien que primitives ont été développées pour permettre l'insertion de nouvelles données comme les plugins ou les images. Ces fonctionnalités étant réservées à des utilisateurs restreints : rôle d'administration ou de développement, il a fallu étudier quels pouvaient être les comportements du système face à de tels cas de figures.

La première idée était de dire qu'à l'insertion de nouvelles images dans la base, l'ensemble des descripteurs disponibles serait aussitôt appliqué sur ces dernières pour la génération des fichiers XML. Deux points importants sont à souligner. En effet, en agissant ainsi, les informations de la base peuvent devenir hétérogènes puisque les images déjà présentes dans la base ne posséderont pas forcément le même ensemble de valeurs de descripteurs, car il est possible que, durant leur insertion, les plugins disponibles ne soient pas les mêmes. Cela ne pose pas de réels problèmes pour le calcul de similarité puisque, comme il est précisé ci-dessus, le calcul est effectué sur le plus grand sous-ensemble commun.

Le deuxième point concerne le temps d'exécution qui peut être observé si le nombre d'images à insérer est conséquent. Si, pour chaque image, on calcule immédiatement les valeurs des descripteurs disponibles, cela peut figer le fonctionnement de la plate-forme pendant une durée importante. Il serait alors judicieux de planifier les calculs par le biais d'une table définissant des heures durant lesquelles les calculs pourraient être lancés en tache de fond de manière automatique, ce qui permettrait également d'harmoniser l'ensemble des informations de la base. Ce type d'outils est bien connu des systèmes Unix, car il existe deux commandes pouvant répondre à ces critères :

– Cron qui est le nom d'un programme permettant aux utilisateurs des systèmes Unix d'exécuter automatiquement des commandes ou des scripts à une date et une heure spécifiées à l'avance.
– At qui est également le nom d'un programme UNIX permettant de définir des tâches à exécuter à une heure précise.

Il suffirait donc d'interfacer ces programmes avec des scripts PHP plutôt que d'exécuter les calculs en temps réel. Cependant, faute de temps, cette fonctionnalité n'a pu être développée.

Altération ou modification de l'image de référence depuis le navigateur

Une autre fonctionnalité intéressante est l'altération en ligne depuis le navigateur internet de l'image de référence de la recherche. Afin, par exemple, d'observer ce que pourrait donner les résultats de la recherche si une partie de l'image seulement était



source de la requête ou bien encore si l'image était légèrement bruitée.

L'implémentation de cette fonctionnalité est un problème complexe puisqu'il souligne le fait que l'application possède deux fonctionnements :
- Ce qui s'exécute du côté client sur le navigateur
- Ce qui s'exécute du côté serveur sur le serveur web

La nature de la requête ayant source chez le client, on peut donc envisager deux approches :
- Exécuter les calculs d'altérations de l'image côté client via une applet et envoyer le résultat au serveur afin que celui-ci puisse recalculer les valeurs des descripteurs afin de relancer la recherche.
- Récupérer toutes les informations relatives à la nature de la requête et les envoyer au serveur pour que ce dernier effectue l'intégralité des traitements de son côté afin de relancer la recherche plus tard.

Quel que soit l'approche sélectionnée, elle n'est pas simple en regard aux contraintes technologiques qu'impose le fonctionnement client/serveur d'une application web.

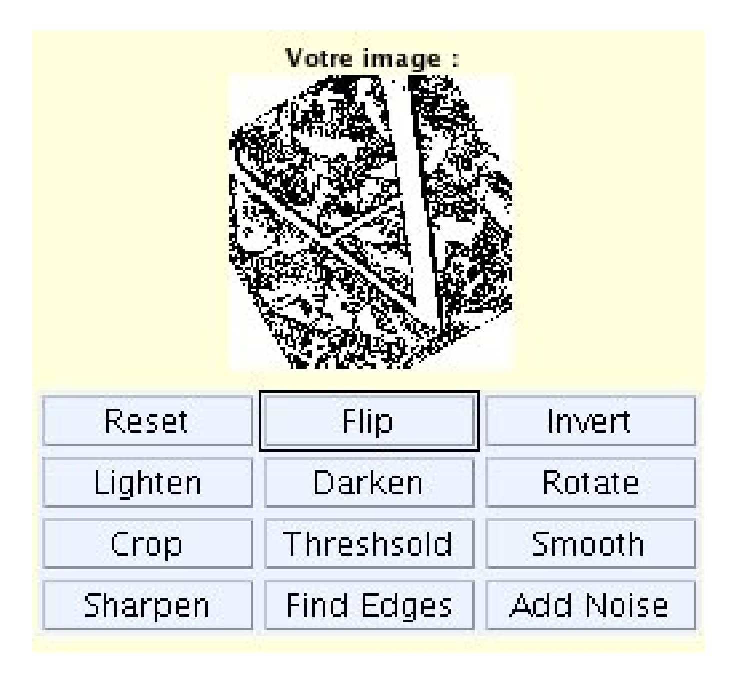


FIG. 5.9 – Applet permettant la modification en ligne de l'image de référence

Une première approche a consisté à utiliser une applet Java développé avec la bibliothèque de traitement d'images *ImageJ* (bibliothèque puissante, rapide et offrant de nombreuses fonctionnalités équivalentes à OpenCV) en l'intégrant à l'interface. Celle-ci fonctionne globalement mais le rapatriement de l'image modifiée n'a pas encore été implémenté, ce qui fait que la recherche ne peut être relancée.



5.6 Authentification

Afin de limiter l'accès de la plate-forme aux seuls utilisateurs autorisés, un système d'authentification basé sur un annuaire LDAP a été réalisé.

Traditionnellement, on définit une liste d'utilisateurs de l'application avec un ensemble de droits, puis l'on rentre ces utilisateurs dans une nouvelle base de données en générale relationnelle. L'originalité du système d'authentification repose sur le principe d'un annuaire, aussi hétérogène soit-il. Mais avant d'en expliquer le choix et ses avantages, nous allons en effectuer une brève présentation.

5.6.1 Présentation de LDAP

cf. Annexe D, page 117

5.6.2 Le choix d'un annuaire LDAP

De plus en plus de grosses et moyennes structures procèdent à la migration de leur catalogue de ressources (personnel, postes, imprimantes, etc...) au sein d'un annuaire de type LDAP. Pourquoi ? Et bien en général parce que cela permet de « centraliser » (nous verrons plus loin pourquoi ce mot peut-être un peu abusif) au sein d'une même base d'informations, des catalogues totalement hétérogènes. Outre cet aspect hétérogène, les annuaires LDAP ont l'avantage de pouvoir s'organiser selon un système hiérarchique de domaines qui peuvent se scinder en plusieurs parties distribuables sur plusieurs serveurs permettant ainsi, par exemple, de répartir physiquement les ressources du catalogue et cela en toute transparence pour les clients LDAP ou bien encore de faire du mirroring (duplication pour sauvegarde et synchronisation). Autre avantage également, les annuaires LDAP possède une grande rapidité d'accès en terme de consultation par rapport aux SGBDs traditionnels, mais cela au détriment d'un accès en écriture plus lent. Enfin LDAP bénéficie également de transmissions sécurisées par le biais d'outils comme le cryptage SSL.

Dans le cadre de ce projet, il parait donc intéressant à terme de pouvoir effectuer l'authentification sur un annuaire regroupant l'ensemble des personnes habilitées à utiliser la plate-forme, sachant que cet annuaire peut être scindé en plusieurs parties (différents laboratoires par exemple). L'idée est, qu'à terme, un annuaire regroupant le personnel soit utilisé avec des informations supplémentaires relatives uniquement à la plate-forme, de sorte qu'il n'y ait pas de duplication de base (une spécifique à l'application en plus des autres bases déjà existantes).

5.6.3 Mise en place de l'annuaire

Afin de préparer cette alternative à un annuaire commun, un petit annuaire LDAP de test a tout de même été créé pour obtenir rapidement une première version fonctionnelle avec une authentification LDAP. Tout comme les solutions web, il existe plusieurs implémentations : OpenLDAP, Active Directory, etc... (plus ou moins rigoureuses de la norme) de LDAP. Sans rentrer dans les détails et en conservant les mêmes choix qui ont été faits pour la solution web LAMP (gratuité, documentation, rapidité

de développement, multi-plate-forme, etc...), OpenLDAP [7] s'est imposé comme étant la solution retenue.

Avant de commencer la configuration du serveur, il est conseillé d'établir l'inventaire des objets et des informations à stocker dans l'annuaire.

Dans le cadre de cette première approche à LDAP, nous avons choisi un ensemble minimaliste d'informations à stocker. Ainsi, les utilisateurs de la plate-forme sont caractérisés par les attributs suivants :
- leur nom (Sn)
- leur prénom (Cn)
- leur mot de passe (UserPassword)
- leur numéro de téléphone (Telephonenumber)
- leur laboratoire d'origine (Title)
- leur groupe d'appartenance comme administrateur, informaticien, non-informaticien. C'est par le biais de ce champs que l'on adaptera l'interface de la plateforme en fonction du groupe d'appartenance.(O)
- leur adresse email (Mail)

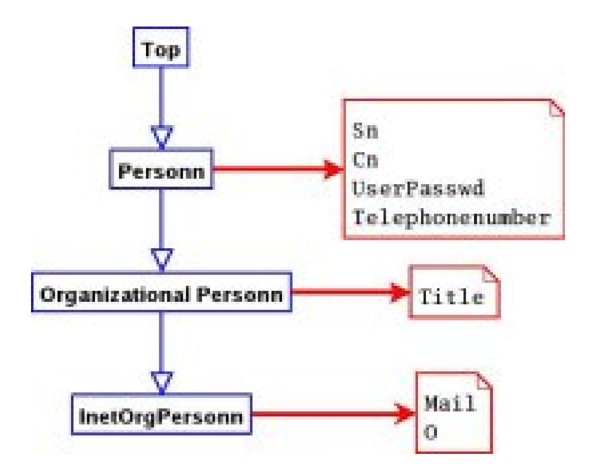


FIG. 5.10 – Schéma LDAP de la plateforme

Les classes d'objets forment ainsi une hiérarchie, au sommet de laquelle se trouve l'objet *top*.

Chaque objet hérite des propriétés (attributs) de l'objet dont il est le fils. On précise la classe d'objet d'une entrée à l'aide de l'attribut *objectClass*. Il faut obligatoirement indiquer la parenté de la classe d'objet en partant de l'objet *top* et en passant par chaque ancêtre de l'objet.

Par exemple, l'objet *inetOrgPerson* a la filiation suivante :

[7]OpenLDAP est un serveur d'annuaire LDAP Open Source et issu des implémentations du protocole par l'Université de Michigan. Il est développé selon les termes de la licence GNU GPL, ce qui signifie qu'il est entièrement gratuit et que son code source est accessible et modifiable.



objectClass : top
objectClass : person
objectClass : organizationalPerson
objectClass : inetOrgPerson

L'objet *person* a comme attributs : *commonName*, *surname*, *description*, *seeAlso*, *telephoneNumber*, *userPassword*.
L'objet fils *organizationalPerson* ajoute des attributs comme : *organizationUnitName*, *title*, *postalAddress*...
L'objet petit-fils *inetOrgPerson* lui rajoute des attributs comme : *mail*, *labeledURI*, *uid* (userID), *photo*...

Une entrée peut appartenir à un nombre non limité de classes d'objets. Les attributs obligatoires sont la réunion des attributs obligatoires de chaque classe.

Une fois que le schéma a été défini dans le serveur OpenLDAP (cf. Annexe B.4.3, page 107), nous devons insérer quelques éléments (ici, des utilisateurs) dans la base. Bien qu'il soit possible de le faire à la main par l'utilisation de ligne de commandes (et par extension, via des scripts), il est bien plus commode d'utiliser une interface graphique pour cette tache. En effet, PhpLdapAdmin, dont l'interface est calqué sur le très célèbre PhpMyAdmin (front-end pour Serveur MySQL), est un front-end écrit en PHP permettant de faciliter nombre de taches d'administration des serveurs LDAP.

5.6.4 Gestion de l'annuaire LDAP

PhpLDAPadmin est un client LDAP web. Il offre une vue hiérarchique, sous forme d'arborescences et des fonctions avancées de recherche, qui rend la navigation et l'administration de l'annuaire très intuitif. Depuis que PhpLdapAdmin est une application web, ce navigateur LDAP fonctionne sur de nombreuses plate-formes, rendant la gestion du serveur LDAP accessible quel que soit le lieu.

Parmi les fonctionnalités intéressantes offertes, on peut citer :
- Ajout, édition et suppression d'entrées de l'annuaire
- Sauvegarde des entrées LDAP (même entre différents serveurs)
- Copie et suppression récursives d'arbres complets
- Possibilité d'effectuer des requêtes LDAP simples ou avancées
- Export d'éléments ou de l'intégralité de l'annuaire au format LDIF et DSML
- Import d'éléments de l'annuaire à partir de fichiers au format LDIF
- Gestion de différents mode de cryptage des mots de passe utilisateur (SHA, Crypt, MD5, Blowfish, MD5Crypt)
- ...

Utilisation de phpLDAPadmin

Comme énoncé précédemment, on peut utiliser l'interface graphique de l'application afin d'insérer un à un chacun des éléments de l'annuaire en cliquant sur *créer* (cf. figure 5.11, page 61), puis en remplissant les différents champs proposés (cf. figure 5.12, page 61).





FIG. 5.11 – Barre de navigation de PhpLDAPadmin

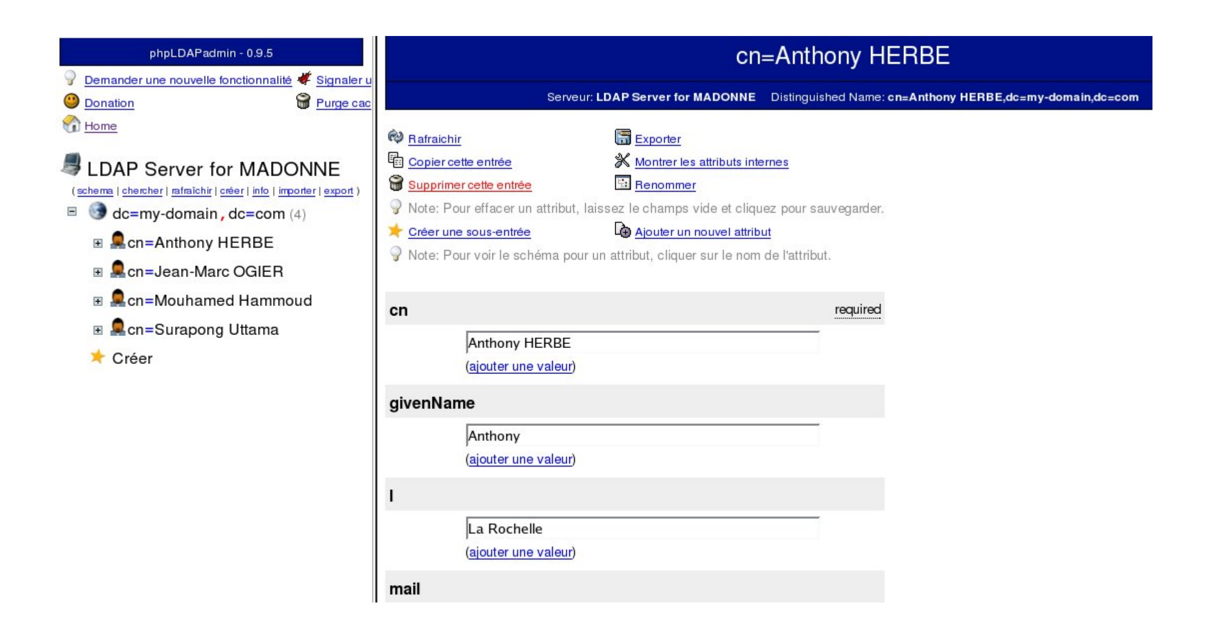


FIG. 5.12 – Edition d'une entrée de l'annuaire dans phpLDAPadmin

Cependant, il est plus rapide (moins d'intervention de la souris) de créer nos entrées à partir d'un fichier qui les regroupe tous. Le chargement dans phpLDAPadmin s'effectue grâce au format standard de LDAP, le format LDIF.

Le Format LDIF

LDAP Data Interchange Format (LDIF) est le standard de représentation des entrées sous format texte. Il est utilisé pour insérer, afficher ou modifier les données de l'annuaire. Le format utilisé est l'ASCII. Toute valeur d'attribut ou tout DN qui n'est pas ASCII est codée en base 64. La forme générale est la suivante :



```
dn: <distinguished name>
objectClass: <object class>
objectClass: <object class>
[...]
attribute type:<attribute value>
attribute type:<attribute value>
[...]
```

Nous avons donc importé le fichier LDIF suivant :

Source 5.2 – Fichier LDIF représentant l'annuaire et ses éléments

```
dn: dc=my-domain,dc=com
dc: my-domain
o: my-domain.com
objectClass: top
objectClass: dcObject
objectClass: organization

dn: cn=Anthony HERBE,dc=my-domain,dc=com
cn: Anthony HERBE
givenName: Anthony
sn: aherbe
o: Labo de La Rochelle
l: La Rochelle
telephoneNumber: 0606060606
mail: anthony.herbe@etudiant.univ-lr.fr
uid: anthony.herbe@etudiant.univ-lr.fr
title: Administrateur
objectClass: top
objectClass: inetOrgPerson
userPassword: {CRYPT}Y1qol5NRZeJL.

dn: cn=Jean-Marc OGIER,dc=my-domain,dc=com
cn: Jean-Marc OGIER
givenName: Jean-Marc
telephoneNumber: 0607080910
mail: jmogier@univ-lr.fr
uid: jmogier@univ-lr.fr
title: Utilisateur
objectClass: top
objectClass: inetOrgPerson
sn: jmogier
userPassword: {CRYPT}cMwtFP06C3nts

...
```

Remarque : les mots de passe cryptés sont obtenus par la ligne de commande *slappasswd -h {crypt}* où *-h* est l'option permettant de préciser le type de hachage.

5.6.5 Interfaçage de l'annuaire avec la plate-forme

Pour utiliser la plate-forme, l'utilisateur doit d'abord s'identifier afin que le système vérifie s'il est habilité ou pas à l'utiliser. Pour cela, la plate-forme doit établir un lien avec le serveur LDAP (qui peut être sur une machine distante) et soumettre une requête de connexion avec un couple comportant l'identifiant et le mot de passe de l'utilisateur préalablement saisi à partir d'une page de la plate-forme.

Pour effectuer ce lien, nous avons utilisé ce qui nous semblait le plus intuitif, à savoir le module LDAP pour PHP ou plus précisément une bibliothèque incluant un ensemble de fonctions écrites en PHP permettant d'effectuer des requêtes sur un annuaire LDAP. Après avoir installé ce module, on bénéficie alors d'une nouvelle API regroupant des fonctions préfixées par *ldap_*.

Séquence d'interrogation avec un serveur LDAP

L'interrogation d'un serveur LDAP avec PHP se fait selon une séquence simple, nécessitant un nombre peu élevé de fonctions spécialisées. La séquence basique est la suivante :

1. Établissement de la connexion avec le serveur LDAP
2. Liaison et authentification sur le serveur (appelé bind en anglais)
3. Recherche d'une entrée (ou bien une autre opération)
4. Exploitation des résultats (uniquement dans le cas d'une recherche)
5. Fermeture de la connexion

Connexion au serveur LDAP

Avant de pouvoir interroger le serveur LDAP, il est essentiel d'initier la connexion. Pour cela l'interpréteur PHP a besoin de connaître quelques renseignements relatifs à l'annuaire :
- L'adresse du serveur
- Le port sur lequel le serveur fonctionne
- La racine de l'annuaire
- Le login de l'utilisateur (généralement root) ainsi que son mot de passe

Source 5.3 – Informations nécessaires à la connexion au serveur LDAP

```
<?
// Fichier de configuration pour l'interface PHP
// de notre annuaire LDAP
$server="indus.univ-lr.fr";
$port="389";
$racine="dc=my-domain, dc=com";
$rootdn="cn=Manager, dc=my-domain, dc=com";
$rootpw="secret";
?>
```



On définit donc cinq variables pour caractériser le serveur LDAP : le nom du serveur, le port (389 par défaut), la racine supérieure de l'arborescence, la chaîne de connexion pour l'administrateur ainsi que son mot de passe.

La première fonction à utiliser est la fonction *ldap_connect()* permettant d'établir une liaison avec le serveur. Sa syntaxe est la suivante :

```
int ldap_connect([string hostname [, int port]])
```

Cette fonction admet en paramètre le nom du serveur (éventuellement le port. Par défaut le port est 389). En cas d'échec cette fonction retourne 0 sinon elle retourne un entier permettant d'identifier le serveur et nécessaire dans les fonctions suivantes.

Voici un exemple de connexion au serveur :

```
<?
echo "Connexion...<br>";
$ds=ldap_connect($server);
?>
```

Toutefois, cette étape n'est pas suffisante pour pouvoir exécuter des opérations sur le serveur LDAP. En effet, il est nécessaire d'initier la liaison (en anglais : to bind) avec le serveur LDAP à l'aide de la fonction *ldap_bind()* dont la syntaxe est la suivante :

```
int ldap_bind (int identifiant [, string bind_rdn [, string
    bind_password]])
```

Cette fonction attend en paramètre l'identifiant du serveur retourné ainsi qu'éventuellement le nom distingué relatif de l'utilisateur (RDN - Relative Distinguished Name) et son mot de passe. Si l'utilisateur et le mot de passe ne sont pas précisés, la connexion se fait de manière anonyme.

La déconnexion du serveur LDAP se fait tout naturellement par la fonction *ldap_close()* avec la syntaxe suivante :

```
int ldap_close (int identifiant)
```

Cette fonction est similaire à la fonction *ldap_unbind()*.



Voici un exemple complet de connexion et de déconnexion à un serveur LDAP :

```
<?
// Fichier de configuration pour l'interface PHP
// de notre annuaire LDAP
$server="indus.univ-l.fr";
$port="389";
$racine="dc=my-domain, dc=com";
$rootdn="cn=Manager, dc=my-domain, dc=com";
$rootpw="secret";

echo "Connexion...<br>";
$ds=ldap_connect($server);

if($ds==1)
{
  // On s'authentifie en tant que super-utilisateur, ici, Manager
  $r=ldap_bind($ds,$rootdn,$rootpw);
  // Ici les opérations à effectuer
  echo "Déconnexion...<br>";
  ldap_close($ds);
}
else{
  echo "Impossible de se connecter au serveur LDAP";
}
?>
```

Grâce à toutes ces directives, nous avons donc pu développer notre procédure d'authentification.

Module d'authentification

Comme la majorité du source de cette plate-forme, un module indépendant a été créé en vu d'encapsuler les méthodes d'interrogation du serveur LDAP afin de rendre l'implémentation transparente par rapport à la bibliothèque utilisée. Le code source en est d'autant plus simplifié.

Voici un extrait du code source situé dans le fichier *index.php* faisant office de page d'accueil et de page d'identification :

Source 5.4 – Source de la procédure d'identification

```
...
$ldap=new LDAP();
if(!empty($login))
{
  if($ldap->login($login,$passwd))
  {
    $_SESSION['auth'] = true;
    print_login_success();
```



```
    }
    else
      print_login_failure();
}
...
```

Comme on peut le voir dans l'extrait ci-dessus, si l'identification est un succès, une variable de session est positionnée permettant de mémoriser l'authentification réussie durant toute la durée de la navigation sur la plate-forme.



FIG. 5.13 – Capture de la page d'identification sur la plate-forme

Pour que l'authentification soit un succès, une fois la présence du serveur LDAP assurée, le programme va s'y connecter en mode super-utilisateur, et rechercher la présence d'un utilisateur ayant comme identifiant celui précisé dans le formulaire de connexion. Dans le cas où il est trouvé, il vérifie la présence et la validité du mot de passe indiqué. Si tout est correcte, l'authentification est assurée et l'utilisateur arrive sur la page de menu de l'application. Sinon, il obtient un message d'erreur l'informant du problème (serveur introuvable, identifiant ou mot de passe erroné).



FIG. 5.14 – Capture de la page du menu principal de la plate-forme



L'utilisateur peut ensuite effectuer les actions en rapport avec ses habilitations.



5.7 Manipulation des arbres XML

La démarche afin de trouver l'outil adéquat pour utiliser des arbres XML au sein de la plate-forme a été la même que pour la partie authentification, à savoir le choix d'un module PHP.

Pour manipuler des données XML, il faut utiliser un parseur de documents XML qui va analyser la structure du document et permettre d'accéder à ses différents composants.

Les parseurs XML s'appuient sur deux modèles possibles de traitements d'un document, connus respectivement sous les acronymes SAX (Simple API for XML) et DOM (Document Object Model). Le premier consiste à parcourir le document linéairement, et à déclencher des fonctions à chaque fois qu'une des catégories syntaxiques (balises ouvrantes, fermantes, texte, instructions de taitement, etc.) constituant un document XML est rencontrée. Le second, quant à lui, s'appuie sur la représentation arborescente du document. Chaque noeud de l'arbre est un objet, doté de méthodes propres au type de noeud, et de pointeurs vers le ou les sous-arbres, le père du noeud, les attributs, etc.

Dans notre cas, l'utilisation d'un parseur de type DOM est donc plus approprié pour créer, éditer et effectuer des requêtes sur un document XML. Pour cela, nous avons utilisé le module DOMXML pour PHP afin de manipuler nos documents, en regard à une API bien fournie (exécution de requête XPath, sauvegarde de documents, transformation d'un fichier XML en un tableau PHP, etc...) et une implémentation simplifiée. Cependant, il faut noter que ce module est EXPERIMENTAL. Cela signifie que le comportement de certaines fonctions et leurs noms peuvent changer dans le temps, cela en vue de l'évolution de la bibliothèque qui tente de se rapprocher de plus en plus de la norme DOM. Néanmoins, les fonctions pouvant poser problèmes sont listées comme étant *deprecated* et des fonctions alternatives sont proposées quand cela est possible. Il suffit donc de bien prendre garde à ne pas les utiliser.

5.7.1 De l'image au document XML

Une fois les descripteurs de l'image calculés par les plugins (cf. Partie 5.5, page 46), il faut pouvoir stocker cette information en vue de la réutiliser ultérieurement pour la recherche de similarités. Nous procédons alors comme ceci :

1. récupération des méta-données et valeurs des descripteurs dans des structures PHP.
2. création d'un DOM vierge avec la bibliothèque DOMXML
3. parcours des structures PHP contenant les données de l'image citées ci-dessus et création et insertion, au fur et à mesure, de ces données dans des noeuds du DOM.
4. suvegarde du DOM ainsi créé dans un fichier XML.

Afin de faciliter le développement et la lisibilité du code, une classe PHP dont voici le prototype, a été créée :



```
<?php
class xmlfile
{
  var $pict;
  var $doc;
  var $file;
  var $xpath;
  ...
  function addnode($parent,$child_label) {...}
  function addnode_withvalue($parent,$child_label,$child_value)
      {...}
  function addnode_withattributes($parent,$child_label,$attributes,
     $attributes_values) {...}
  ...
  function search_subtree_from($element_type) {...}
  ...
  function create_from_picture($picture,$xml_dir="") {...}
  function load_from_xmlfile($filename,$xml_dir="") {...}
  function save_doc($freetextannotation="",$creationtime="") {...}
  ...
}
?>
```

Comme on peut l'observer sur le prototype ci-dessus, trois méthodes préfixées par *addnode* permettent l'ajout de noeuds avec ou sans attributs ou valeurs dans le document *$doc* de l'instance de la classe. Trois autres méthodes permettent respectivement la création, le chargement et la sauvegarde de documents XML et une dernière ajoute, quant à elle, le support de la recherche dans le document à partir d'un élément par le biais de requêtes XPath.

L'utilisation de cette classe rend transparente la manière de manipuler les données XML. Ainsi, si le besoin de changer d'implémentation se fait sentir, il suffira de changer le comportement des méthodes sans pour autant modifier toute l'application.

5.7.2 Le défaut de l'implémentation actuelle

La sauvegarde des documents XML de l'application s'effectue par le biais de fichiers XML, ce qui constitue cependant un inconvénient majeur pour l'indexation et la recherche des informations.

En effet, à chaque image est associé un fichier XML contenant l'intrégalité des informations qui la décrivent (méta-données et descripteurs) ; ce qui a pour effet que la base de données n'est autre qu'un ensemble de fichiers XML. Ainsi, pour effectuer une recherche de similarités entre images, on est obligé d'ouvrir l'intégralité de ces fichiers de manière séquentielle pour récupérer l'information nécessaire à la comparaison. Il est donc impossible d'accélérer la recherche en effectuant la recherche sur un sous-ensemble de documents.



Néanmoins, cette implémentation ne devait être qu'une étape afin de pouvoir obtenir une première version fonctionnelle de l'application. La solution retenue à terme est l'utilisation d'une base de données XML native, qui répond mieux à nos besoins en terme d'indexation puisqu'il suffit, tout comme dans un SGBDR traditionnel, d'effectuer une requête XQuery sur la base plutôt que d'ouvrir tous les fichiers de la base. Une base XML native est en fait considérée comme une forêt constituée des arbres de chacune des images.

Une fois le développement de la classe PHP citée ci-dessus effectué, une tentative d'implémentation de solution à partir de base XML native, a débuté. Pour cela, et selon les conseils d'un collaborateur, la base eXist a été choisie. Une fois celle-ci installée et configurée, quelques fonctions d'interrogation de cette base ont été développées en utilisant la bibliothèque Query-eXist (existant en Perl et surtout en PHP).

Voici un exemple de fonctionnement :

Source 5.5 – Exemple de requête de sélection sur base eXist depuis PHP

```
<?PHP
  include('eXist/eXist_soap.php');

  $user="madonne_admin";
  $password="adm_exist";
  $wsdl="http://l3iexp.univ-lr.fr:8080/exist/services/Query?wsdl";

  $db = new eXist_SOAP($user,$password,$wsdl);
  $db->setDebug();
  if(!$db->getError())
    $db->connect();
  else
    die($db->getError());

  $query="for \$signatures in //VisualDescriptor return \$signatures
     ";

  $result=$db->xquery($query) or die($db->getError());

  $hits=$result["HITS"];
  $queryTime=$result["QUERY_TIME"];
  $collections=$result["COLLECTIONS"];

  print "<br />\nfound $hits hits in $queryTime ms.\n";

  // XQuery Result
  print "<p><b>Result of the XQuery:</b></p>";
  print "<pre>";
  print(htmlspecialchars(implode("", $result["XML"])));
  //print (implode("", $result["XML"]));
  print "</pre>";

  $db->disconnect();
?>
```



Le script PHP ci-dessus permet de se connecter à la base eXist et d'afficher l'intégralité des sous-arbres *//VisualDescriptor* de la forêt. On peut effectuer une analogie avec le fonction d'une requête SQL comme *SELECT * FROM*. On peut donc très facilement sélectionner n'importe quel sous-arbre dans l'ensemble de la forêt. De plus, comme avec le langage SQL, la clause *WHERE* permet de filtrer les noeuds rendus par la clause FOR. Le contenu de la clause *WHERE* doit être booléen. Par exemple, si on désire récupérer l'ensemble des valeurs du descripteur Zernike de toutes les images de la base, il faudra procéder ainsi :

```
FOR $signatures in //VisualDescriptor
WHERE $signatures[@type="Zernike"]
RETURN $signatures/value
```

Cependant, beaucoup de temps a été perdu sur la compréhension de l'installation et la configuration de eXist (mise en place d'un serveur Tomcat nécessaire). N'ayant pu trouvé (faute encore de temps) comment insérer des éléments dans la base eXist et n'ayant pu pousser trop la recherche (toujours faute de temps), l'implémentation de la solution à base d'eXist a été stoppée.

5.7.3 Comparaison des arbres XML

Pour obtenir une mesure de similarité entre une image et une autre, il faut calculer la distance entre chaque signature commune au deux images. Concrêtement, en utilisant les données XML, cela revient à :

1. Parcourir l'arbre XML de chaque image et selectionner le plus petit sous-ensemble commun de descripteur.
2. Comparer un à un chacun des noeuds de chaque sous-ensemble et effectuer une distance locale.
3. Puis calculer la distance globale à partir des distances locales.

Cependant, si on laisse l'algorithme de calcul tel quel, on ne tient pas compte de l'influence que peuvent prendre certains descripteurs par rapport à d'autres. Il est donc nécessaire de pondérer chaque distance locale avec un « poids ». Le calcul de la distance globale qui a été implémentée est la distance pondérée suivante :

$$Local_Dist_i = \sqrt{\sum_{k=0}^{n} (v_{a_k} - v_{b_k})^2}$$

$$Global_Dist = \frac{\sum_{i=0}^{m} (p_i * Local_Dist_i)}{\sum_{i=0}^{m} p_i}$$

où les v_{a_k} (respectivement v_{b_k}) correspondent aux différentes valeurs décrivant le descripteur k pour l'image a (respectivement b).



5.8 Réseau et calcul distribué

Il a été démontré précédemment l'intérêt d'effectuer du calcul distribué dans le cadre du calcul des signatures, que ce soit dans un souci de réactivité ou bien à cause de contraintes techniques.

Le principe de fonctionnement que nous souhaiterions obtenir pourrait se résumer à ceci :

1. Le serveur web désire effectuer un ou plusieurs calculs de signatures sur un serveur de calculs
2. Il effectue sa demande de calculs au sein d'un message destiné au serveur de calculs. Ce message devra logiquement contenir :
   – Des informations sur la commande à exécuter pour le calcul
   – Des informations sur l'image sur laquelle les calculs devront être effectués
3. Le serveur de calculs reçoit le message
4. Les calculs s'effectuent sur le serveur de calculs
5. Le serveur de calculs renvoie les résultats des calculs au serveur web

Faire interagir des programmes différents en réseau a toujours été une affaire complexe, notamment si on souhaite standardiser certains aspects de cette interaction (une sorte de couche à mi-chemin entre la couche « Transport » et la couche « Application » du modèle OSI, comme la défunte couche 5).

Les Web Services sont un ensemble de protocoles qui permettent, au moins sur le papier, de faire communiquer (avec un protocole de haut niveau, pas juste des bits) des programmes tournant sur des machines différentes et écrits dans des langages de programmation différents.

Ils permettent donc de connecter différents composants du système d'information (y compris entre organisations différentes).

Un des modèles le plus simple en programmation distribuée est sans conteste RPC. RPC (Remote Procedure Calling) est un protocole permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'application. Ce protocole est utilisé dans le modèle client-serveur et permet de gérer les différents messages entre ces entités. Ainsi, notre serveur web devient le client RPC et notre serveur de calculs, le serveur.

Néanmoins, les messages qui transiteront devront être envoyés à partir de l'application web hébergée par notre serveur web et transiter via le protocole HTTP (éventuellement HTTPS si l'on désire sécuriser les messages RPC). Une bonne solution est d'utiliser le format XML pour encapsuler les messages et les transmettre au sein de trame HTTP.
Nous obtenons ainsi un protocole couplant à la fois XML et RPC.

Il existe bien évidemment d'autres Web Services que XML-RPC, comme par exemple

SOAP [8], qui possède également de nombreux avantages, mais ce dernier peut rapidement devenir complexe dans sa mise en oeuvre. C'est pourquoi nous lui préférerons XML-RPC qui répond totalement à nos besoins, malgré sa simplicité d'implémentation et ses quelques lacunes (Pas normalisé sous un organisme neutre (IETF, W3C, etc...), ce qui est un handicap pour être utilisé comme base pour d'autres protocoles normalisés).

5.8.1 La technologie XML-RPC

Issue de la société "Userland Software" (Avril 1998), la technologie XML-RPC repose, comme son nom l'indique, sur XML (Extensible Markup Language) et sur le protocole RPC (Remote Procedure Calling). Tandis qu'XML apporte l'indépendance vis-à-vis de la plate-forme d'exécution, RPC apporte la possibilité d'effectuer "des appels de procédures" via internet.

Grâce à l'utilisation de la technologie XML-RPC, il est ainsi possible à différentes applications de dialoguer entre elles sans se soucier des systèmes sur lesquels elles fonctionnent, ni même du langage dans lequel elles ont été écrites.

En effet, il existe de nombreuses implémentations de cette technologie : PHP mais aussi Perl, Python, C/C++, Java, .Net, Ruby, Rebol, Cold Fusion... La liste est longue.

Fonctionnement

Notre application étant écrite en PHP, une bibliothèque XML-RPC écrite dans ce langage a été utilisée. Voici donc le fonctionnement de cette bibliothèque.

XML-RPC : un dialogue client/serveur

Nous allons « disséquer » le code d'un client Php, mais pour nous connecter sur quel serveur ? Bonne question. Celui-ci est construit sur la bibliothèque initialement développée par Edd Dumbill (Useful Information Company) avant d'être, depuis sa version 1.0, ouverte à un développement plus collectif via SourceForge.

Pourquoi utiliser une bibliothèque alors que Php 4.1 propose désormais par défaut des fonctions XML-RPC ? Bien que celles-ci soient écrites en C et offrent donc de bonnes performances, le site php.net prévient l'utilisateur potentiel de ces fonctions que celles-ci peuvent à tout moment changer de nom, on peut lire : « Be warned and use this extension at your own risk ».

Au regard de cette recommandation, il est bien évidemment déconseillé de les utiliser en production.

[8]SOAP (Simple Object Access Protocol) est un protocole de RPC orienté objet bâti sur XML. Le transfert se fait le plus souvent à l'aide du protocole HTTP, mais peut également se faire par un autre protocole, comme SMTP. Il a été initialement défini par Microsoft et IBM, mais est devenu depuis une recommandation du W3C.



La bibliothèque PHPXML-RPC (http://phpxmlrpc.sourceforge.net/) va donc grandement nous faciliter le travail. Il suffit par la suite de connaître les différents mécanismes et échanges nécessaires au dialogue du client avec le serveur pour rapatrier et exploiter les résultats transmis par le serveur. N'oublions pas bien sûr la documentation de l'API qui nous renseignera sur les différentes méthodes disponibles.

Puisque le serveur est fourni, voici ce qu'il nous reste à faire :

1. Créer un objet « client », c'est l'initialisation du client
2. Créer un message à destination du serveur
3. Envoyer celui-ci
4. Le réceptionner (vérifier les codes retour) puis exploiter le résultat obtenu.

Première étape : initialisation du client
Nous voulons initialiser notre client, nous allons donc tout naturellement utiliser la classe prévue à cet effet : *xmlrpc_client*.
La documentation est claire, voici la syntaxe pour créer notre objet client :

```
$client=new xmlrpc_client($server_path,$server_host,$server_port);
```

Le premier paramètre est le chemin du script qui va gérer la requête XML-RPC, le second paramètre est le nom (ou l'IP) du serveur, quant au troisième (facultatif), il représente le port sur lequel nous nous connecterons (80 par défaut).

Au niveau des méthodes fournies par cette classe, on trouve :
- *send()* : son nom est suffisamment explicite, nous l'utiliserons à la prochaine étape
- *setCredentials* : Transporte le login/password à des fins d'authentification HTTP
- *setCertificate* : Permet d'employer HTTPS. Attention, dans ce cas Php doit être compilé avec l'extension *curl*. De plus, PHP 4.0.2 au moins est nécessaire pour faire fonctionner HTTPS, à noter également qu'un bogue de la version 4.0.6 empêche son utilisation.
- *setDebug* : Très utile pour obtenir des informations retournées par le serveur.

Deuxième étape : construction de la requête
Cette étape repose sur l'utilisation de la classe *xmlrpcmsg* dont voici la syntaxe :

```
$message=new xmlrpcmsg($methodName,$parameterArray);
```

$methodName contient le nom de la méthode à appeler et *$parameterArray* contient un tableau de ses paramètres.

La technologie XML-RPC permet de manipuler 8 types de données :



– int
– double
– string
– boolean
– base64
– dateTime.iso8601
– array
– struct

Nous ne sommes pas obligés d'indiquer les paramètres éventuels d'une méthode lors de la déclaration de l'objet, il est possible de le faire ultérieurement en utilisant la méthode *addParam()* après la création de celui-ci.

Nous décidons de passer néanmoins les paramètres de la méthode lors de la création de l'objet. Peu importe le moment où nous le faisons, il faut de toute façon avoir recours à la classe *xmlrpcval* afin d'encapsuler nos données dans un format compréhensible par toutes les autres classes de la bibliothèque.

Voici trois exemples issus de la documentation de cette classe qui permettent de comprendre comment utiliser nos paramètres :

```
$myInt=new xmlrpcval(1267,"int");
$myString=new xmlrpcval("Hello, World!","string");
$myBool=new xmlrpcval(1,"boolean");
```

Fort de ces exemples, et des précédents paragraphes, nous résumons donc la syntaxe de la déclaration de notre message à :

```
$message=new xmlrpcmsg("compute_signature",
                       array(new xmlrpcval("/signatures_bin_path/
                          compute_sig1","string"),
                             new xmlrpcval("/pictures_path/img1.bmp"
                                 ,"string")));
```

Troisième étape : l'envoi du message

La syntaxe de cette étape se base sur la méthode *send()* de la classe *xmlrpc_client* que nous évoquions tout à l'heure.

Récapitulons les deux étapes précédentes :

```
$client=new xmlrpc_client('/path/xml-rpc_server.php','compute-server
    .univ-lr.fr',80);
$message=new xmlrpcmsg("compute_signature",
                       array(new xmlrpcval("/signatures_bin_path/
                          compute_sig1","string"),
                             new xmlrpcval("/pictures_path/img1.bmp"
                                 ,"string")));
```



Il est temps d'envoyer notre message au serveur :

```
$resultat=$client->send($message);
```

La syntaxe exacte de cette étape est la suivante :

```
$resultat=$client->send($message,$timeout,$server_method);
```

Lorsqu'ils sont omis, les paramètres *$timeout* et *$server_method* prennent respectivement les valeurs 0 (pas de timeout) et *HTTP*.

Quatrième étape : analyse du code retour et exploitation des résultats

C'est un objet de type *xmlrpcresp* qui nous est renvoyé. Si celui-ci est égal à zéro, la connexion au serveur n'a pas pu s'effectuer. Si ce dernier n'est pas égal à zéro, il se peut quand même qu'un problème soit survenu : le serveur peut ne pas avoir compris notre demande. Il faut donc s'assurer si la méthode *faultcode()* renvoie zéro, signe qu'aucune erreur n'est survenue. En cas d'erreur, la méthode *faultString()* renvoie un descriptif de l'erreur commise.

Laissons parler le code pour une version plus synthétique de ces deux étapes :

```
if(!$resultat)
{
print "<p>Could not connect to HTTP server.</p>";
}
elseif($resultat->faultCode())
{
print "<p>XML-RPC Fault #".$resultat->faultCode().": ".
$resultat->faultString();
}
```

Si tout s'est bien passé, il faut exploiter les données renvoyées par le serveur. La méthode *value()* permet de « traduire » le résultat retourné par le serveur en un objet de type *xmlrpcval*. Afin de récupérer de manière exploitable par PHP les valeurs renvoyées par la méthode *compute_sig1* (par ex.), il nous faut passer par plusieurs étapes : une fois la méthode *value()* appliquée, il faut ensuite extraire de la valeur obtenue les valeurs « réelles ». Cela donne :

```
$xmlrcp_obj=$resultat->value(); // traduction en objet xmlrpcval
/* Extraction et conversion de l'objet
xmlrpc de telle façon que PHP puisse
le manipuler des types classiques */
```

```
$real_values=php_xmlrpc_decode($xmlrcp_obj);
```

5.8.2 Implémentation de XML-RPC dans la plate-forme Madonne

Afin de rendre le code source plus modulaire et réutilisable, une classe PHP du nom de *my_xmlrpc_client* a été créée par mes soins encapsulant l'ensemble des fonctions du client XML-RPC. Voici le code qui lui est associé :

Source 5.6 – Client XML-RPC écrit en PHP

```
<?php
if(eregi("xmlrpc_client.php",$_SERVER['PHP_SELF'])) die();
require_once("xmlrpc.inc");

/** Client XML-RPC.
*   Classe permettant l'envoi et la réception
*   de messages provenant d'un serveur XML-RPC.
**/
class my_xmlrpc_client
{
var $client;
var $debug_mode;

function my_xmlrpc_client($path,$host="localhost",$port=80,
    $debug_mode=0)
{
    // Make an object to represent our server.
    $this->client=new xmlrpc_client($path,$host,$port);

    // Bénéficier du mode debug
    $this->client->setDebug($debug_mode);
    $this->debug_mode=$debug_mode;
}

function sendmsg($function,$params)
{
    if(!is_string($function)) die("Le nom de fonction doit être une
        chaîne de caractères");
    // Send a message to the server.
    $message = new xmlrpcmsg($function,array(php_xmlrpc_encode(
        $params)));
    if($this->debug_mode==1)
        highlight_string($message->serialize());
    return $this->client->send($message);
}

function test_received_msg($result)
{
    // Process the response.
    if(!$result)
    {
```

```
        print "<p>Could not connect to HTTP server.</p>";
        return FALSE;
    }
    elseif($result->faultCode())
    {
        print "<p>XML-RPC Fault #".$result->faultCode().": ".$result
            ->faultString();
        return FALSE;
    }
    else
        return TRUE;
}

function unpack_values_from_valid_msg($msg)
{
    return php_xmlrpc_decode($msg->value());
}
}
?>
```

Pour l'implémentation du serveur, nous utiliserons l'ensemble des méthodes fournies par la bibliothèque PHP décrite ci-dessus. Le constructeur de la classe du serveur effectue en grande partie tout le nécessaire :

```
function exec_cmd($m)
{
...
}

$server=new xmlrpc_server(array("execute"=>array("function"=>"
    exec_cmd"));
```

Le seul argument du constructeur est un tableau associatif qui associe aux noms de méthodes du serveur les noms des fonctions du script. Lors de la réception d'une requête, celle-ci est analysée puis dispatchée à la fonction associée, responsable du retour de l'objet *xmlrpcresp*, qui fournit une valeur de retour « sérialisée » à l'appelant (le client XML-RPC).
Voici plus en détails, ce que la méthode *exec_cmd* devrait effectuer :

```
function exec_cmd($m)
{
global $EXIT_FAILURE;
$xmlrpcval = $m->getParam(0);
if($xmlrpcval->kindOf()=="scalar"
   and $xmlrpcval->scalartyp()=="string")
{
    $cmd = $xmlrpcval->scalarval();
    ob_start();
    passthru($cmd,$result_exec);
```

```
    $result=array(new xmlrpcval($result_exec,'int'),
                  new xmlrpcval(ob_get_contents(),'string'));
    ob_end_clean();
}
else
    $result=array(new xmlrpcval($EXIT_FAILURE,'int'),
                  new xmlrpcval("Le contenu du message doit être une
                      chaîne de caractère",'string'));
return new xmlrpcresp(new xmlrpcval($result,"array"));
}
```

Tout d'abord, une vérification en ce qui concerne le typage du message de la requête doit être effectuée afin d'être sur que le message reçu du client est bien formé. Puis, tout comme pour le client, l'objet du message doit être converti en un type exploitable par PHP. Enfin, le résultat, qu'il s'agisse d'une erreur ou pas, est encapsulé à son tour dans un objet XML-RPC (via *new xmlrpcval(...)*) afin d'être renvoyé au client XML-RPC sous la forme d'un objet *xmlresp*.

Une seconde méthode de contrôle consiste en l'utilisation de signatures associées aux méthodes du serveur. En effet, chaque signature renseigne sur le nombre et le type des paramètres de chaque méthode ainsi que le type de valeur retournée. Ainsi, si une signature a été définie, le serveur effectue une vérification à l'appel de la méthode concernée :

```
$exec_cmd_sig=array(array($xmlrpcArray,$xmlrpcString));
function exec_cmd($m)
{
...
return new xmlrpcresp(new xmlrpcval($result,"array"));
}

$server=new xmlrpc_server(array("execute"=>
                          array("function"=>"exec_cmd",
                                "signature"=>$exec_cmd_sig)))
                                   ;
```

Tout comme pour notre client, une classe XML-RPC a été développée en surcouche de celle fournie par la bibliothèque afin de faciliter toutes modifications ultérieures de l'implémentation du code. Voici donc le code PHP associé à cette classe :

Source 5.7 – Serveur XML-RPC écrit en PHP

```
<?php
require_once("xmlrpc.inc");
require_once("xmlrpcs.inc");
$EXIT_SUCCESS=0;
$EXIT_FAILURE=1;

$exec_cmd_sig=array(array($xmlrpcArray, $xmlrpcString));
$exec_cmd_doc='Execute a command line in a shell. Accepts a string,
    and return the result of the command line';
```



```
function exec_cmd($m)
{
    global $EXIT_FAILURE;
    $xmlrpcval=$m->getParam(0);
    if($xmlrpcval->kindOf()=="scalar"
    and $xmlrpcval->scalartyp()=="string")
    {
        $cmd=$xmlrpcval->scalarval();
        ob_start();
        passthru($cmd,$result_exec);
        $result=array(new xmlrpcval($result_exec,'int'),
                      new xmlrpcval(ob_get_contents(),'string'));
        ob_end_clean();
    }
    else
        $result=array(new xmlrpcval($EXIT_FAILURE,'int'),
                      new xmlrpcval("Le contenu du message doit être une chaîne de caractère",'string'));
    return new xmlrpcresp(new xmlrpcval($result, "array"));
}

$server=new xmlrpc_server(array("execute"=>array("function"=>"exec_cmd",
                                                  "signature"=>$exec_cmd_sig,
                                                  "docstring"=>$exec_cmd_doc)));
?>
```



5.9 Installation et Déploiement

Une fois la plate-forme fonctionnelle sur l'environnement de développement, c'est à dire un portable s'exécutant sur la distribution Linux Fedora Core 3, il a fallu refaire le point sur l'ensemble de l'architecture de la plate-forme avant de l'installer en production.

Voici donc les schémas de son fonctionnement et de ce dont elle a besoin :

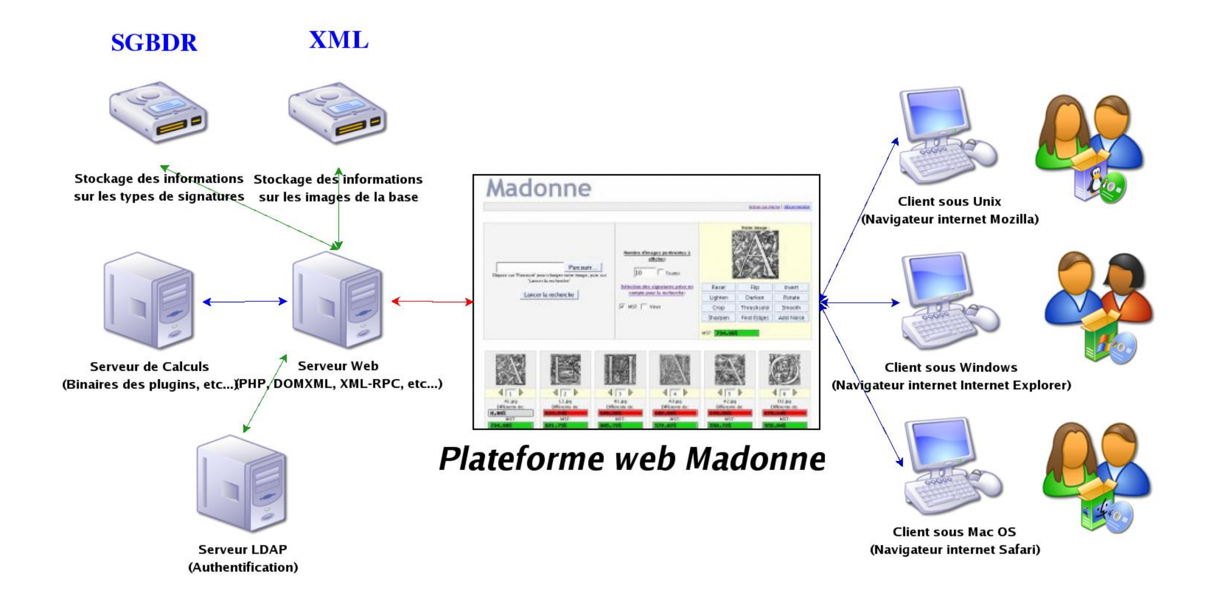


FIG. 5.15 – Schéma du fonctionnement générale de la plate-forme

Ce premier schéma illustre la répartition des différentes machines et bases de données ainsi que les interactions entre les différents acteurs du système global.



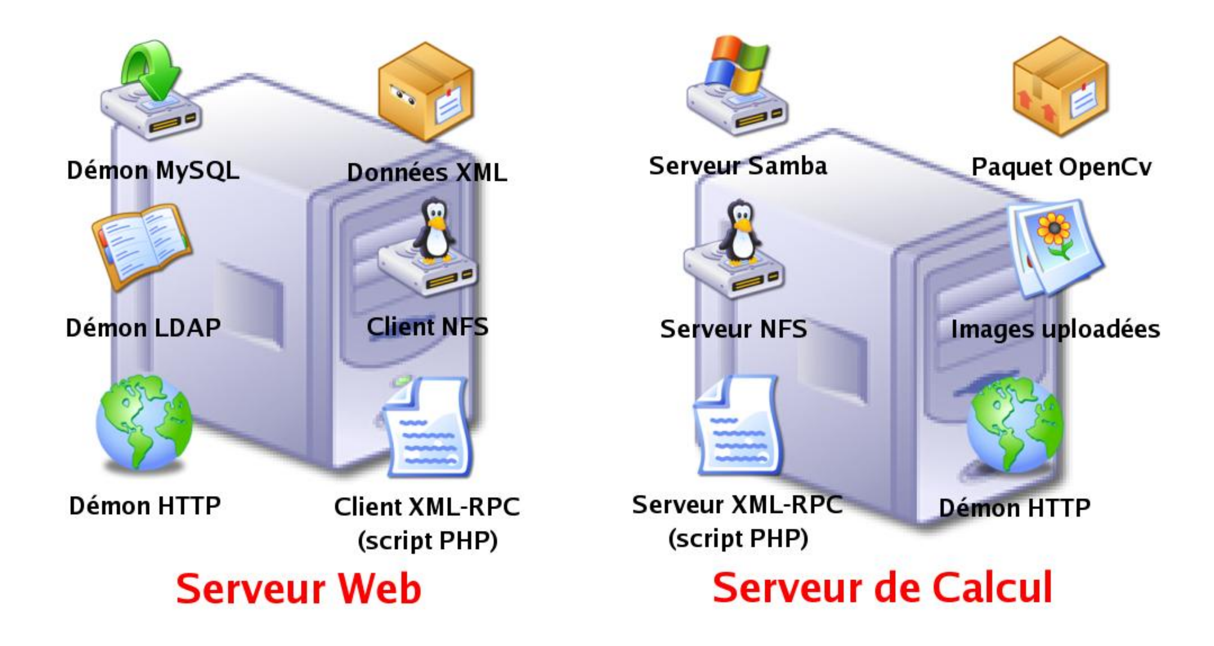


FIG. 5.16 – Répartition des différents éléments de la plate-forme sur les serveurs

Cette représentation illustre l'organisation des différents éléments situés sur les différentes machines de la plate-forme (ressources fichiers, exécution des démons, applications tierces installées, etc.).





FIG. 5.17 – Ensemble des modules nécessaires à Apache pour le fonctionnement de la plate-forme

Enfin, le dernier schéma recense l'intégralité des modules installés sur le serveur Apache.

5.9.1 Passage en pré-production

Afin de tester la procédure de migration d'un environnement de développement bien rodé par la période du stage, il a été décidé d'effectuer un premier passage de l'application dans un environnement dit de « pré- »production. Ceci afin d'éviter de se retrouver dans une situation inconfortable et inattendue pouvant nuire aux données ou à la configuration de l'environnement de production. Il m'a été offert (selon les différentes éventualités) d'installer la totalité des éléments de la plate-forme sur un serveur bi-processeur (qui servira à terme uniquement de serveur de calculs). Ce dernier s'exécutant sur une distribution Linux Red Hat Entreprise version 3 (RHE3), il m'a fallu d'abord comprendre le fonctionnement de cette dernière n'étant moi-même habitué qu'à des distributions grand public.

En effet, la RHE3 étant dédiée à des machines de type « serveurs », celle-ci ne bénéficie pas des paquets (fichiers rpms) auxquels j'étais habitué. Pour la bonne et simple raison que seul Red Hat est habilité à maintenir des paquets officiels dits stables. Pour effectuer une installation sur ce type de distribution, il a fallu d'abord souscrire à des



listes de paquets disponibles via une interface dans un navigateur internet, élément totalement inhabituel dans les distributions grand public. Après avoir assimilé le principe du système RHE3, je me suis heurté à des problèmes de mise à jour du serveur sur lesquels j'ai perdu encore énormément de temps. Après avoir trouvé une parade à ces problèmes, essentiellement dus à une mauvaise configuration du serveur que je n'ai pu solutionner (ils existaient avant mon arrivé et demeurent toujours), j'ai donc pu installer et mettre à jour l'ensemble des modules pour Apache, PHP, etc. ainsi que les différentes bases de données et la bibliothèque OpenCV.

Malgré une configuration presqu'identique des installations de l'environnement de pré-production (RHE3) et celui de développement (FC3), le fonctionnement de la plate-forme semble différé pour une raison inconnue. En effet, bien que l' application semble fonctionner parfaitement sur les deux environnements, la méthode d'authentification a un comportement anormal uniquement sur la RHE3.

Le point qui diffère se produit lorsqu'on tente de s'identifier avec un identifiant valide mais avec un mot de passe invalide. Alors que l'authentification est refusée sur la FC3, ce qui est le fonctionnement normal, elle est acceptée sur la RHE3. Je n'ai à ce jour pas identifié l'origine du problème qui n'est cependant pas important d'en un premier temps puisque cela s'est produit sur l'environnement de pré-production et pas sur celui de production. Tous les autres cas d'utilisation possèdent par contre un comportement normal quel que soit l'environnement.

Une fois les tests effectués et validés sur la plate-forme de pré-production, la migration vers l'environnement de production a pu débuter.

5.9.2 Passage en production

L'environnement de production est un poste de travail classique s'exécutant sur une Fedora Core 1, distribution Linux grand public semblable à celle de développement quoi que plus ancienne. La migration s'avérait délicate puisque certaines données sensibles y étaient stockées (certains sites internet du laboratoire, etc. . .) et il a fallu prendre en compte les configurations existantes et éviter à tout prix de rentrer en conflit avec les configurations mises en place.

Cette machine sert en fait uniquement de serveur web, le reste restant sur le serveur de calculs (machine de pré-production). Il a été nécessaire de s'occuper uniquement de la partie serveur web, le reste ayant déjà été effectué sur le serveur de calcul lors du passage en pré-production. Parmi les éléments qui étaient nécessaire à la plateforme et qui étaient déjà installés et plus ou moins configuré, figuraient :
- Un serveur web Apache en version 2
- Le module PHP pour Apache dans une version 4
- Le module MySQL pour PHP

Il a alors fallu rajouter les modules manquants (cf. Figure 5.17, page 83) ainsi qu'effectuer les configurations adéquates.

Un point important concerne la configuration du serveur Apache sur laquelle je n'ai pu effectuer toutes les modifications qui s'imposaient en terme de sécurité. En

effet, depuis les versions 4.1 de PHP, la variable d'environnement *register_globals* est à *OFF* (depuis la découverte d'un trou de sécurité) alors qu'elle ne l'était pas avant. Le développement de la plate-forme l'a donc pris en compte dès le début en utilisant les méthodes de programmation PHP associées. Cela n'étant pas forcément le cas pour les applications existant sur le serveur web et ne souhaitant pas interférer dans le bon fonctionnement de ces applications, je n'ai donc pas modifié la valeur de cette variable dans la configuration de la machine de production.

Système de fichiers des machines de production

La plate-forme Madonne manipulant un certain nombre de ressources (fichiers et dossiers), il a fallu vérifier que chacune possédait les droits adéquats. Parmi ces ressources, on retrouve :

1. les fichiers et dossiers propres à l'interface web de la plate-forme autres que sa configuration
2. les fichiers de systèmes concernant le stockage de la base de données MySQL et LDAP
3. les fichiers de configuration des démons Apache, MySQL, LDAP
4. les répertoires où sont stockés les images, les fichiers XML et les plugins

Pour que la plate-forme fonctionne correctement, il faut :

1. que l'utilisateur nommé *apache* (sous FC et RHE) puisse lire, parcourir l'ensemble des dossiers et fichiers relatifs à l'interface et la configuration de la plate-forme.
2. qu'il puisse écrire dans les dossiers relatifs au cache de l'application (par défaut dossier nommé *cache*), dans les dossiers où sont stockés les images uploadées, les fichiers XML et les plugins.
3. que le serveur de calcul puisse lire le contenu des répertoires cités ci-dessus.

Par défaut, l'intégralités des dossiers et fichiers auront pour propriétaire et pour groupe l'utilisateur *apache* avec des droits de lecture, d'écriture et d'exécution associés, le reste des droits devant rester en lecture seule. Voici ce que donne cette configuration :

```
drwxrwxr-x  2 apache apache   4096 jun 21 22:39 cache
drwxrwxr-x  2 apache apache   4096 jun  1 19:21 cgi-bin
drwxrwxr-x  2 apache apache   4096 jun 14 18:40 classes
-rw-rw-r--  1 apache apache    392 avr  5 12:12 clean.php
-rw-rw----  1 apache apache   1601 jun 21 22:32 config.php
-rw-rw----  1 apache apache   2105 jun 21 22:38 env_vars.php
drwxrwxr-x  2 apache apache   4096 jun 14 18:43 functions
-rw-r--r--  1 apache apache 960428 avr  7 22:09 ij.jar
drwxrwxr-x  4 apache apache  36864 mai 27 20:06 images
-rw-rw-r--  1 apache apache    826 jun 14 18:42 index.php
drwxrwxr-x  2 apache apache  12288 mai 14 20:58 layers
-rw-rw-r--  1 apache apache    105 mar  4 00:52 logout.php
-rw-rw-r--  1 apache apache    782 avr  5 18:48 menu.php
-rw-rw-r--  1 apache apache   6262 jun  1 18:30 search.php
drwxrwxr-x  2 apache apache   4096 jun  2 16:38 templates
-rw-rw-r--  1 apache apache   4475 jun  1 18:30 upload.php
-rw-rw-r--  1 apache apache   3969 avr 25 21:39 upload_sig.php
-rw-r--r--  1 apache apache   1333 mai 27 18:37 xmlrpc_client.php
-rw-r--r--  1 apache apache  50512 mai 17 21:39 xmlrpc.inc
-rw-r--r--  1 apache apache  13243 mai 17 21:39 xmlrpcs.inc
```

Néanmoins, il faut prendre garde au droit de lecture des fichiers relatifs à la configuration de l'application (*env_vars.php* et *config.php*) puisqu'ils contiennent des don-



nées sensibles comme des mots de passe, des identifiants et des adresses IP. Leur droit de lecture ne doit donc être donné qu'exclusivement à l'utilisateur *apache*.

Partage de ressources

Nous venons de voir que certaines ressources comme des répertoires doivent être accessibles par deux machines (celle de calcul et celle hébergeant le serveur web principal). Pour se faire, des partages ont été créés entre les deux machines sur ces dits répertoires. La méthode utilisée la moins restrictive fut l'élaboration de partage de type NFS. Ainsi, l'intégralité des ressources partagées a été physiquement placée sur le serveur de calcul, il a suffit ensuite d'y installer et configurer un serveur NFS et de donner l'accès en lecture et/ou en écriture uniquement à l'autre machine afin de sécuriser l'accès. Sur l'autre machine, les dossiers ont été « montés » comme n'importe quel système de fichiers réseau en utilisant la table des montages, ce qui rend l'ensemble transparent pour la plate-forme qui n'est pas « consciente » de la spécificité de ces ressources.

Pour faciliter l'accès aux ressources comme les plugins, les images et les couches associées, des partages SAMBA ont également été mis en place, ceux-ci ne fonctionnant bien évidemment qu'en réseau local.

Chapitre 6

Bilan du stage

Le projet aura été très riche en expériences, que ce soit dans les domaines purement technique, organisationnel, scientifique ou tout simplement relationnel.

6.1 Évaluation du travail réalisé

Au fur et à mesure de l'avancement du projet, l'ampleur de ce dernier a pu être comparé à un iceberg. Parti sur les bases (bien que faibles) du travail effectué au cours de l'UE Vision 2, le travail avait été divisé en un nombre de domaines bien définis. Chacun de ces domaines a pu être abordé et des solutions ont été proposées. L'objectif que je m'étais fixé était d'obtenir une plate-forme fonctionnel et utilisable. L'objectif a donc été atteint.

Cependant, le travail effectué n'est que la partie émergeante du projet, car nombre de points n'ont pu être abordés faute de temps et/ou de moyens humains. L'ensemble du travail effectué a également pour ambition d'être réutilisable afin que le projet puisse aboutir, si bien évidemment le développement continue.

6.2 Problèmes rencontrés

Comme dans tout projet, nombre de difficultés ont vu le jour, que ce soit en lien direct avec le projet ou bien dans le cadre de travail. Comme cela a été signalé dans ce rapport, il a été nécessaire de travailler sur des postes coupés physiquement du réseau du laboratoire du moins pour l'environnement de développement. Cela a débuté avec mon ordinateur personnel sur lequel tout avait été préparé et bien entamé. Malheureusement, des problèmes matériels sont survenus m'obligeant à l'envoyer au service après-vente. Le temps de retrouver un autre poste dans les mêmes conditions (réinstallation de tout l'environnement) a été un préjudice supplémentaire en terme de temps.

6.2.1 Les problèmes organisationnels

Etant donné l'importance du projet en terme de travail à effectuer, il a parfois été obligatoire et très frustant de devoir abandonner des solutions qui paraissaient évidentes à terme, par manque de temps que ce soit en terme de développement ou bien en terme d'apprentissage de nouvelles technologie (ex. : Base de données XML native

sur eXist). Des choix ont donc été faits en ce qui concerne l'organisation du développement de la plate-forme, ils sont le fruit de décisions prises en complète autonomie, que celles-ci fussent bonnes ou mauvaises.

6.2.2 Les problèmes techniques

Dans le domaine purement technique ou plutôt technologique, le fait de s'imprégner de technologies naissantes et souvent synonymes de longues études, recherche de documentations, méthodologies empiriques. Ce fut le cas pour toute la partie « données » basée sur le XML légèrement orienté pour une approche du MPEG-7.

Enfin, la partie « traitement d'images » n'a pas été non plus un travail d'une grande facilité puisqu'il fallait composer entre les avancées et les difficultés de chacun dans le domaine surmonté des contraintes liées au caractère « web » de la plate-forme.

6.2.3 Les problèmes relationnels

Ce fut loin d'être les plus inquiétants du moins au niveau du laboratoire L3i où la coopération fut grande, que ce soit dans les moyens techniques ou encore dans les réflexions scientifiques. Cependant, nombreuses observations ont montré l'énorme fossé de la communication entre les laboratoires. En effet, il fut très difficile de se faire écouter des scientifiques, et ce encore plus si on n'a pas le statut minimum de thésard. Nombre de mes requêtes concernant la communication ont été ignorées. Tout comme moi, je suppose que les différents intervenants du consortium avaient d'autres priorités que de répondre à mes courriers électroniques. Bien qu'ayant fait quelques efforts de relance, cette tache a été replacée après les autres en partie par faute de temps. Pour appuyer mon affirmation, j'ai pu observer au cours du Workshop Madonne auquel j'ai pu participer (et qui m'a énormément intéressé) là encore un malaise concernant la communication : point souligné par un des clients majeurs du consortium, un intervenant du CESR (si mes souvenirs sont corrects), qui s'est étonné que les 3/4 des ressources aient été utilisées sur des travaux sur les lettrines alors que le contexte de Madonne se situe sur l'intégralité des documents anciens, toute forme confondue.

6.3 Estimation du travail restant à effectuer

Comme je l'ai fait remarquer plus haut, le reste de l'iceberg commence juste à devenir visible. Une grande quantité de travail est à prévoir. On peut cependant en souligner certains points comme :

- l'intégration d'un véritable système d'indexation couplé à une base de données XML native comme eXist qui est à mon avis une très bonne piste.
- l'implémentation d'un système d'apprentissage (approche statistique et/ou réseaux neuronaux) concernant le relevance feedback.
- une amélioration générale de l'interface.
- la continuité de l'intégration de l'annuaire LDAP.
- etc...

Conclusion

Les objectifs d'un stage professionnel sont multiples : le stagiaire doit tirer une double expérience (immersion dans le monde du travail et acquisition de nouvelles connaissances sur les activités professionnelles) et pouvoir apporter à la structure qui l'a accueilli un bénéfice aussi bien sous forme de nouvelles compétences liées à sa formation qu'à sa personnalité.

Ce stage a été enrichissant, aussi bien au niveau humain que professionnel, et sera un atout pour mon entrée dans la vie active. Il m'a apporté de nouvelles connaissances tant organisationnelles que techniques et m'a permis d'approfondir les compétences que j'ai acquises tout au long de ma scolarité.

Le projet a mis en évidence les nombreux domaines abordés par un projet d'une telle envergure et l'importance du découpage, de la planification, de la répartition et la synchronisation des taches au sein d'une équipe.

Le développement d'une plate-forme générique permettant d'évaluer la pertinence de traitements d'images est un projet bien ambitieux. J'espère avoir pu apporter ma pierre à l'édifice et que l'ensemble formé par ce rapport et la plate-forme développée pourra constituer une première étape vers un système très prometteur dans les années à venir.

Liste des annexes

Annexe A

Manuel d'utilisation de la plate-forme Web Madonne

A.1 Prérequis

Pour utiliser la plate-forme web, il faut posséder un navigateur internet disposant d'une machine virtuelle JAVA (nécessaire au chargement d'un plugin). L'utilisateur doit également faire partie de la liste des personnes habilitées à l'utiliser. Pour cela, elle doit posséder un compte sur l'annuaire LDAP de la plate-forme.

A.2 Mode de fonctionnement

Une fois en possession d'un identifiant et d'un mot de passe valide, il est possible d'accéder à la plate-forme à cette adresse http://l3iexp.univ-lr.fr/madonne_web/ (que ce soit en intranet ou en internet). Il arrive alors face à la page d'authentification :



FIG. A.1 – Page d'accueil de la plate-forme

Si une erreur de saisie concernant l'identifiant ou le mot de passe se produit, le message suivant s'affichera (un autre du même type s'affichera signalant un problème avec le serveur LDAP) :





FIG. A.2 – Erreur d'identification

Si l'identification s'est déroulée avec succès, l'utilisateur accède au menu de l'application :



FIG. A.3 – Menu principal de la plate-forme

A partir de cet écran, selon les droits de l'utilisateur, il peut :
- effectuer une recherche,
- insérer une ou plusieurs images dans la base,
- insérer un ou plusieurs plugins dans la base,
- supprimer les ressources de la plate-forme (images et informations associées)

Si l'utilisateur sélectionne *insérer un ou plusieurs images/plugins*, il obtiendra les écrans suivants :



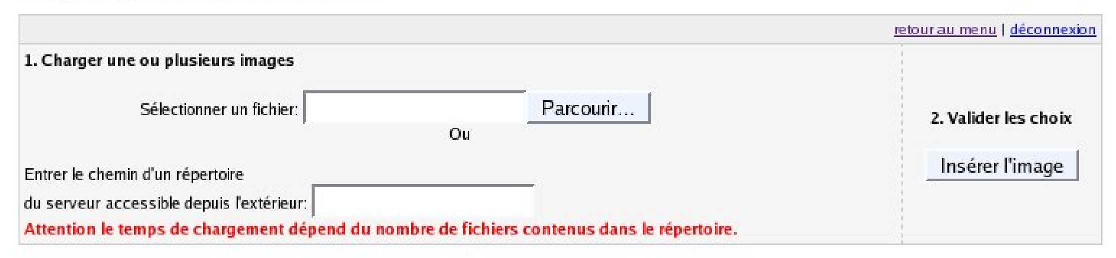


FIG. A.4 – Menu d'insertion d'images

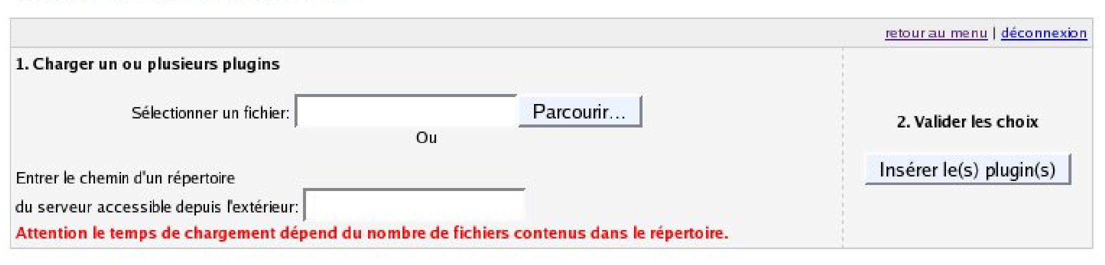


FIG. A.5 – Menu d'insertion de plugins

Les fonctionnalités de ces deux écrans sont identiques. Elles permettent d'insérer un fichier situé sur le disque de l'utilisateur ou encore à partir d'un chemin "absolu" du serveur pointant sur un répertoire rempli de fichiers.

Une fois, l'opération validée, l'utilisateur est ramené au menu principal précédent.

Si l'utilisateur sélectionne l'action *Effectuer une recherche*, il est aussitôt redirigé ver cette page :



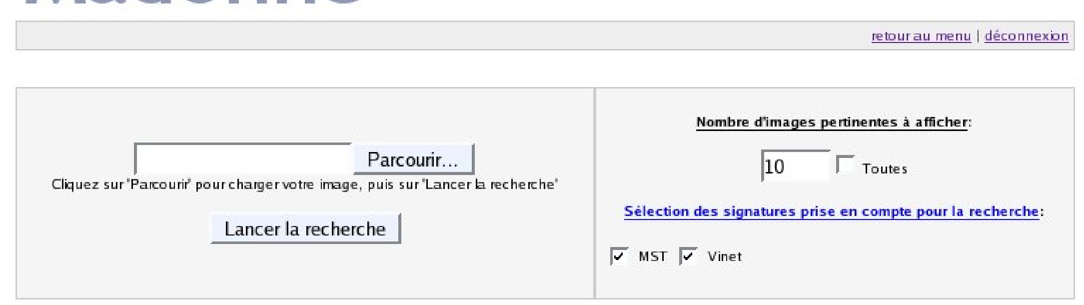


FIG. A.6 – Menu de sélection de l'image de référence

Sur cette écran, l'utilisateur peut sélectionner le nombre d'images qui apparaîtront dans le résultat et les descripteurs à prendre en compte lors de la recherche.

Si tout se déroule normalement, l'écran de résultat est affiché (sinon un écran signal l'erreur apparue (problème de connexion au serveur de calcul, aucune image dans la base, etc...)) :



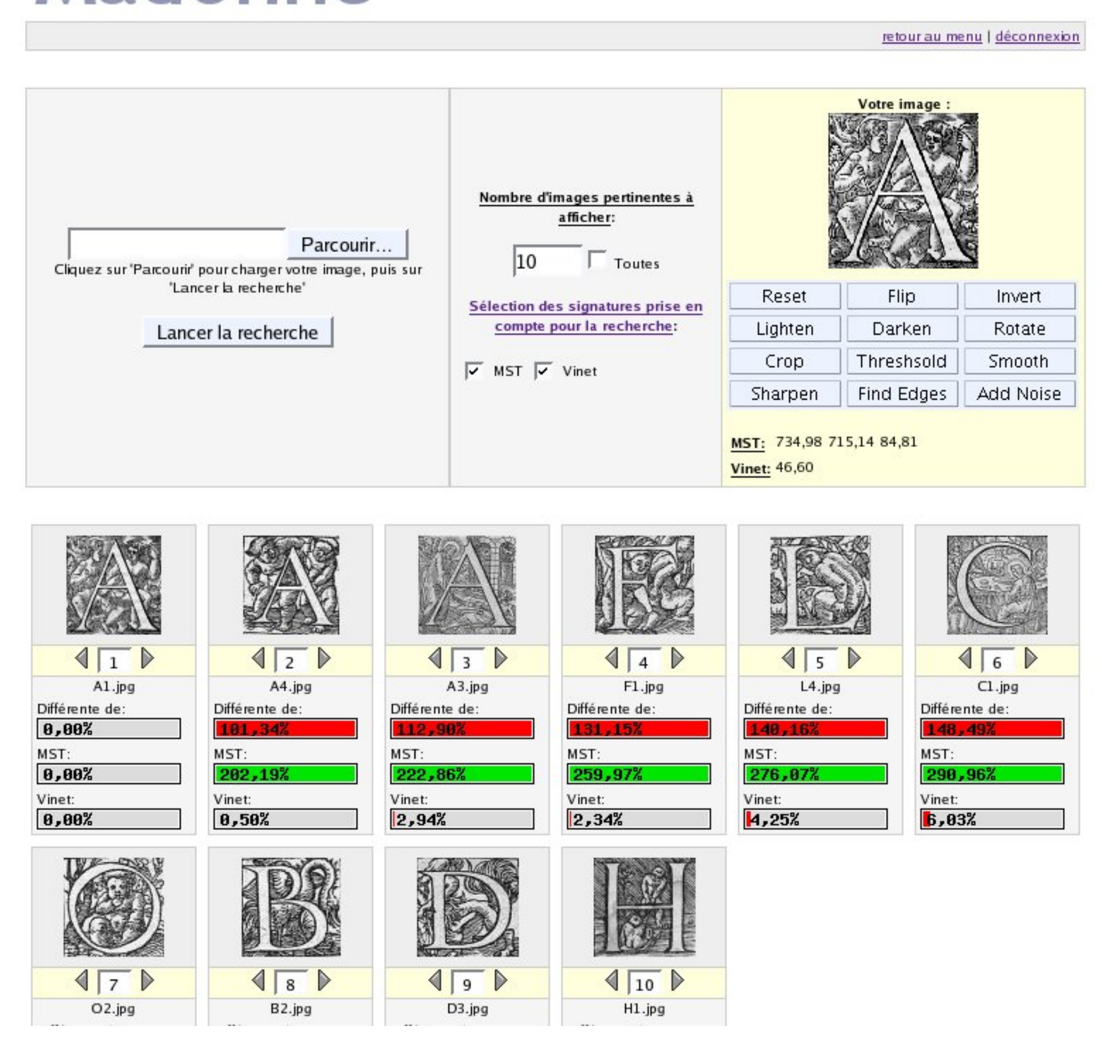


FIG. A.7 – Résultat de la recherche d'images

L'utilisateur peut intervenir sur l'image de référence en utilisant les boutons de l'applet JAVA chargée :



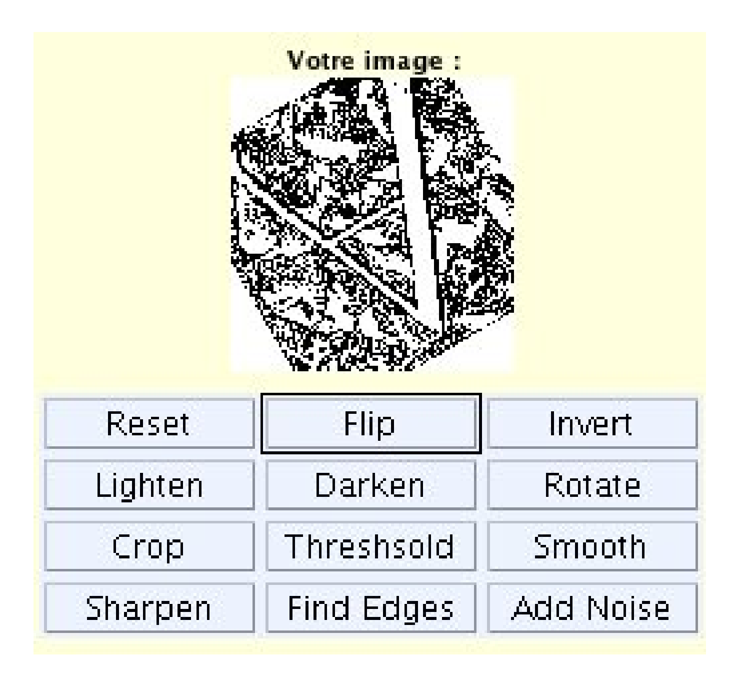


FIG. A.8 – Edition de l'image de référence

Sinon, des informations supplémentaires autres que l'affichage des distances globales et locales peuvent être affichées en cliquant simplement sur l'image désirée :

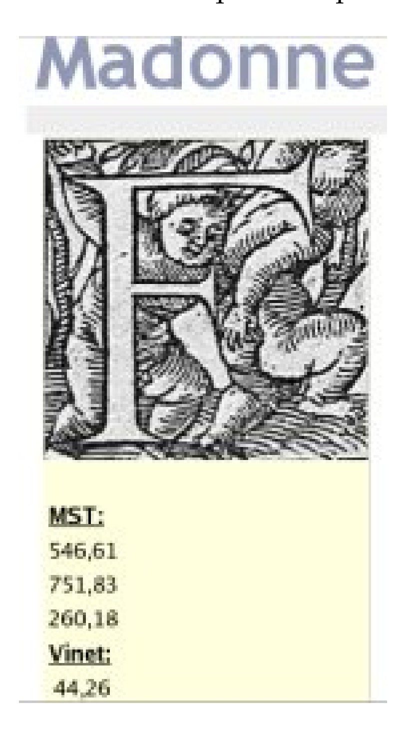


FIG. A.9 – Information détaillée d'une image

Enfin, l'utilisateur peut relancer la recherche à partir de la même image mais avec des paramètres de recherche différents sans pour autant avoir à resélectionner l'image.

A.2.1 Remarques diverses

Il peut arriver que la plate-forme s'éternise sur une recherche, cela est du à la surcharge du serveur de calculs qui est monopolisé à ce moment précis par des calculs demandant beaucoup de ressources.

Annexe B

Administration et installation du système

Ce chapitre a pour but de faciliter l'administration, la maintenance et la mise en place de la plate-forme puisqu'il est en quelque sorte un journal de bord retraçant mon expérience dans ce domaine dans le cadre de ce projet.

B.1 Des commandes utiles

Afin de savoir ce qui doit être nécessairement installé sur une machine, il faut pouvoir récupérer le maximum d'informations la concernant. Toutes les commandes qui seront citées par la suite fonctionnent sur un environnement Linux quel qu'il soit (en général).

Tout d'abord, il est toujours utile de connaître les commandes équivalentes aux outils que l'on utilise habituellement en mode graphique car il peut arriver que l'on se retrouve sans mode graphique parce que le serveur X est corrompu engendrant une situation délicate si l'on manque de connaissances dans l'administration en lignes de commandes.

La première commande que j'ai utilisée pour savoir ce qui était installé sur les différentes machines de la plate-forme est celle-ci :

```
rpm -qa
```

Cette commande permet de lister tous les paquets installés sur la machine. Comme seulement certains nous intéressent, il faut rediriger la sortie de la commande dans une commande de filtrage comme la commande *grep* et en utilisant des jokers (caractère *). Ainsi si je veux savoir si apache est installé et connaître sa version, j'exécute la commande suivante :

```
rpm -qa | grep apache
```

Cela fonctionne pour la version 1 d'*apache* mais pas pour la version 2 puisque le paquet a été renommé en *httpd*, il faut donc effectuer ceci :

```
rpm -qa | grep httpd
```

De même si on veut connaître l'ensemble des paquets installés pour *php*, on exécutera la commande :

```
rpm -qa | grep php
```



Ainsi, pour que la plate-forme fonctionne, on aura besoin des paquets suivants (avec des versions plus ou moins proches) :

Source B.1 – Paquets à installer

```
# rpm -qa|grep http&&rpm -qa|grep php&&rpm -qa|grep mysql&&rpm -qa|
    grep ldap&&rpm -qa|grep opencv
httpd-2.0.52-3.1
php-gd-4.3.11-2.5
php-ldap-4.3.11-2.5
php-mysql-4.3.11-2.5
php-4.3.11-2.5
php-domxml-4.3.11-2.5
mysql-3.23.58-16.FC3.1
mysql-server-3.23.58-16.FC3.1
openldap-clients-2.2.13-2
openldap-servers-2.2.13-2
openldap-2.2.13-2
opencv-0.9.6-1
```

Pour installer un paquet en ligne de commande, il existe plusieurs façons et cela dépend de la distribution Linux utilisée. Dans notre cas, nous nous concentrerons sur la Fedora Core et la Red Hat Entreprise. Dans les deux cas, on peut utiliser la commande :

#rpm -Uvh nomfichier.rpm

Mais ceci uniquement dans le cas où l'on possède déjà le paquet à installer. L'inconvénient est le fait que la commande *rpm* ne résout pas les dépendances et refusera d'installer un paquet si vous n'avez pas tout les bibliothèques qu'il requiert. Pour éviter cela on peut utiliser sous Fedora, la commande :

yum install php-gd* php-domxml* php-ldap* opencv*

Celle-ci va chercher directement sur Internet le ou les paquets désirés ainsi que ses dépendances (pour peu que yum soit installé).

Avec la Red Hat Entreprise, cela diffère du fait qu'il faut planifier l'installation de paquet via leur site internet. Une fois la planification validée, vous pouvez exécuter # rhn_check pour synchroniser la planification avec le système.

B.1.1 Autres commandes

Il est toujours pénible de ne pas trouver l'emplacement d'un fichier sur une machine. Heureusement, sous Linux, il existe plusieurs fonctions de recherche. L'une d'entre elles se nomme *locate*. Pour qu'elle fonctionne correctement, il faut mettre à jour l'index des fichiers en root via la commande *updatedb*. Une fois fini, vous pouvez rechercher un fichier ou un répertoire de cette façon : locate fichier .



B.2 Installation de OpenCV

En ce qui concerne OpenCV, je conseille de l'installer à partir des sources compressées dans un tar.gz pour être sûr d'avoir la bonne version. Pour cela vous pouvez effectuer :

```
# wget http://ovh.dl.sourceforge.net/sourceforge/opencvlibrary/
    opencv-0.9.6.tar.gz
# tar xvzf opencv-0.9.6.tar.gz
# cd opencv-0.9.6
# ./configure
# make
# su
# make install
```

Après avoir fait *./configure*, vérifiez bien à la fin que vous avez toutes les bibliothèques nécessaires (hormis peut-être *raw1394* ou *dc1394* qui posent parfois problèmes à la compilation d'Opencv). Voici grossièrement les paquets nécessaires :

```
# rpm -qa|grep libpng-&&rpm -qa|grep libjpeg-&&rpm -qa|grep libtiff
    -&&rpm -qa|grep gtk&&rpm -qa|grep ffmpeg&&rpm -qa|grep 1394
    libpng-devel-1.2.8-1.fc3

libpng-1.2.8-1.fc3
libjpeg-devel-6b-33
libjpeg-6b-33
libtiff-3.6.1-10.fc3
libtiff-devel-3.6.1-10.fc3
gtk2-devel-2.4.14-3.fc3
gtk2-2.4.14-3.fc3
ffmpeg-0.4.9-7_cvs20050418.rhfc3.at
ffmpeg-devel-0.4.9-7_cvs20050418.rhfc3.at
libraw1394-1.2.0-0_5.rhfc3.at
libraw1394-devel-1.2.0-0_5.rhfc3.at
libdc1394-1.0.0-3.rhfc3.at
#
```

Une fois OpenCV compilé et installé via *make* et *make install*, éditez le fichier */etc/ld.so.conf* et ajoutez la ligne */usr/local/lib*, sauvegardez puis lancez *ldconfig* toujours en root.

Afin de faciliter la compilation des futurs programmes écrits avec OpenCV, je conseille fortement d'ajouter un fichier *opencv.pc* pour l'utilitaire *pkg-config*. Pour cela, créez en root le fichier */usr/local/lib/pkgconfig/opencv.pc* et copiez ce qui suit dedans :

```
# Package Information for pkg-config

prefix=/usr/local
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${prefix}/include/opencv
```



```
Name: OpenCV
Description: Intel(R) Open Source Computer Vision Library
Version: 0.9.6
Libs: -L${libdir} -lcxcore -lcv -lhighgui -lcvaux
Cflags: -I${includedir}
```

Ceci vous permettra de compiler vos programmes de la manière suivante :

```
g++ `pkgconfig --cflags` -c mon_fichier.cpp
g++ `pkgconfig --libs` -o mon_executable mon_fichier.o
```

B.3 Configuration de Apache et PHP

La configuration du serveur HTTPD se situe dans le fichier */etc/httpd/conf/httpd.conf*, les lignes ayant de l'intérêt pour la plate-forme Madonne sont :

```
...
# Pour activer le support de LDAP dans Apache
LoadModule ldap_module modules/mod_ldap.so
...
# Utiliser l'encodage européen de caractères
AddDefaultCharset ISO-8859-1
...
```

Pour la configuration de PHP, il faut éditer le fichier */etc/php.ini* et regarder les valeurs suivantes qui peuvent être modifiées suivant les besoins :

```
...
# Resource Limits
max_execution_time = 30     ; Maximum execution time of each script,
     in seconds
max_input_time = 60     ; Maximum amount of time each script may
    spend parsing request data
memory_limit = 8M      ; Maximum amount of memory a script may
    consume (8MB)
...
# Maximum allowed size for uploaded files.
upload_max_filesize = 2M
...
```

Une fois que tout est installé et configuré, on peut relancer tous les démons utilisé par l'application :



```
# /sbin/service httpd restart
# /sbin/service mysqld restart
# /sbin/service ldap restart
# /sbin/service smb restart
# /sbin/service nfs restart
```

B.4 Installation et configuration de la plate-forme

Pour installer la plate-forme, copiez sur le serveur web l'intégralité des fichiers dans l'emplacement */var/www/html/madonne_web* et faite de même pour le serveur de calculs. Ensuite, effacez les fichiers et répertoires inutiles de façon à ce que l'organisation des fichiers ressemble à celle-ci :

Pour le serveur web :

```
drwxr--r-x  2 apache apache   4096 jun 21 22:39 cache
drwxrwxrwx  2 apache apache   4096 jun  1 19:21 cgi-bin
drwxrwxrwx  2 apache apache   4096 jun 14 18:40 classes
-rw-rw----  1 apache apache   1601 jun 21 22:32 config.php
-rw-rw----  1 apache apache   2105 jun 21 22:38 env_vars.php
drwxrwxrwx  2 apache apache   4096 jun 14 18:43 functions
-rw-r--r--  1 apache apache 960428 avr  7 22:09 ij.jar
drwx---r-x  4 apache apache  36864 mai 27 20:06 images
-rw-rw-rw-  1 apache apache    826 jun 14 18:42 index.php
drwxrwxr-x  2 apache apache  12288 mai 14 20:58 layers
-rw-rw-rw-  1 apache apache    782 avr  5 18:48 menu.php
-rw-rw-rw-  1 apache apache   6262 jun  1 18:30 search.php
drwxrwxrwx  2 apache apache   4096 jun  2 16:38 templates
-rw-rw-rw-  1 apache apache   4475 jun  1 18:30 upload.php
-rw-rw-rw-  1 apache apache   3969 avr 25 21:39 upload_sig.php
-rw-r--r--  1 apache apache   1333 mai 27 18:37 xmlrpc_client.php
-rw-r--r--  1 apache apache  50512 mai 17 21:39 xmlrpc.inc
-rw-r--r--  1 apache apache  13243 mai 17 21:39 xmlrpcs.inc
```

Pour le serveur de calculs :

```
drwxr--r-x  2 apache apache   4096 jun 21 22:39 cache
drwxrwxrwx  2 apache apache   4096 jun  1 19:21 cgi-bin
drwx---r-x  4 apache apache  36864 mai 27 20:06 images
drwxrwxr-x  2 apache apache  12288 mai 14 20:58 layers
-rw-r--r--  1 apache apache   1333 mai 27 18:37 xmlrpc_server.php
-rw-r--r--  1 apache apache  50512 mai 17 21:39 xmlrpc.inc
-rw-r--r--  1 apache apache  13243 mai 17 21:39 xmlrpcs.inc
```



Ensuite, il faut créer les partages concernant les ressources communes aux deux serveurs. Pour cela on va configurer le serveur NFS situé sur le serveur de calculs qui hébergera les données et paramétrer le client NFS sur le serveur web.

B.4.1 Partage NFS

Partie serveur

Les 3 fichiers de configuration principaux sont */etc/exports*, */etc/hosts.deny* et */etc/-hosts.allow*.

/etc/exports

Le fichier */etc/exports* est très simple :

```
/var/www/html/madonne_web/cache indus(rw,sync)
/var/www/html/madonne_web/images indus(rw,sync)
/var/www/html/madonne_web/layers indus(rw,sync)
/var/www/html/madonne_web/cgi-bin indus(rw,sync)
```

Cela signifie que l'on autorisera uniquement la machine indus à accéder à tout ces répertoires en lecture et écriture (rw).

/etc/hosts.deny

On va interdire toutes les machines qui ne sont pas autorisées explicitement dans le */etc/hosts.allow*. Un bon vieux "ALL : ALL" interdira l'accès à tous les services à partir de toutes les machines. On peut cependant être plus précis en écrivant :

```
portmap:ALL
lockd:ALL
mountd:ALL
rquotad:ALL
statd:ALL
```

/etc/hosts.allow

Dans le même esprit que pour le */etc/hosts.deny*, ce fichier a l'architecture suivante :
`[service]: [IP de la machine client]`

Donc pour autoriser moca à se connecter à un partage NFS, on écrira :

```
portmap:moca
lockd:moca
mountd:moca
```



```
rquotad : moca
statd : moca
```

On va pouvoir lancer les services : # /sbin/service nfs start
La commande *rpcinfo -p* permet de vérifier que les services fonctionnent. Elle devrait produire un résultat dans cet esprit :

```
# rpcinfo -p
program vers proto   port
100000    2   tcp     111  portmapper
100000    2   udp     111  portmapper
100024    1   udp     737  status
100024    1   tcp     739  status
100011    1   udp     851  rquotad
100011    2   udp     851  rquotad
100003    2   udp    2049  nfs
100003    2   tcp    2049  nfs
100005    1   udp     872  mountd
100005    2   udp     872  mountd
100005    1   tcp     875  mountd
100005    2   tcp     875  mountd
#
```

Pour recharger les services NFS (par exemple après une modification du fichier de configuration) : # /sbin/service nfs restart
le serveur est prêt !

Partie client

Pour utiliser NFS v3, il faut au minimum la version 2.10m du programme *mount*. Pour voir sa version, taper : # mount −V

On va maintenant pouvoir monter nos partages !
identifiez vous sur la machine *indus* en tant que root et faites :

```
# mount moca:/var/www/html/madonne_web/cache /var/www/html/
    madonne_web/cache
# mount moca:/var/www/html/madonne_web/cgi-bin /var/www/html/
    madonne_web/cgi-bin
# mount moca:/var/www/html/madonne_web/images /var/www/html/
    madonne_web/images
# mount moca:/var/www/html/madonne_web/layers /var/www/html/
    madonne_web/layers
#
```

En principe tout devrait bien se dérouler.
Pour monter ces partages définitivement à chaque démarrage de la machine, éditons



le fichier */etc/fstab* de la manière suivante :

```
...
moca:/var/www/html/madonne_web/cache /var/www/html/madonne_web/cache
    nfs rw 0 0
moca:/var/www/html/madonne_web/cgi-bin /var/www/html/madonne_web/cgi
    -bin nfs rw 0 0
moca:/var/www/html/madonne_web/images /var/www/html/madonne_web/
    images nfs rw 0 0
moca:/var/www/html/madonne_web/layers /var/www/html/madonne_web/
    layers nfs rw 0 0
```

Et voilà, le tour est joué !

B.4.2 Configuration et installation de la base de données MySQL

Après avoir installé MySQL (qui en général déjà présent sur les serveurs), il faut créer la base et les tables. Pour cela nous allons utiliser un outil fort pratique déjà installé sur les machines, phpMyAdmin.

Ouvrez dans un navigateur phpMyAdmin avec les droits d'administration pour pouvoir créer une base et des tables, puis créez la base *madonne_web*. Une fois cela effectué, cliquez sur la base nouvellement créée et sélectionnez *SQL* pour pouvoir exécuter une requête SQL. Dans le champ texte, entrez ce qui suit puis validez :

```
CREATE TABLE `images` (
  `id` int(11) NOT NULL auto_increment,
  `path` varchar(255) NOT NULL default '',
  `xml_path` varchar(255) NOT NULL default '',
  PRIMARY KEY (`id`),
  UNIQUE KEY `xml_path` (`xml_path`),
  UNIQUE KEY `path` (`path`)
) TYPE=MyISAM AUTO_INCREMENT=75 ;

INSERT INTO `images` VALUES (38,'/var/www/html/madonne_web/images/A1
    .jpg','./images/xml/A1.xml');
INSERT INTO `images` VALUES (39,'/var/www/html/madonne_web/images/A2
    .jpg','./images/xml/A2.xml');
INSERT INTO `images` VALUES (40,'/var/www/html/madonne_web/images/A3
    .jpg','./images/xml/A3.xml');
INSERT INTO `images` VALUES (41,'/var/www/html/madonne_web/images/A4
    .jpg','./images/xml/A4.xml');
INSERT INTO `images` VALUES (42,'/var/www/html/madonne_web/images/B1
    .jpg','./images/xml/B1.xml');
INSERT INTO `images` VALUES (43,'/var/www/html/madonne_web/images/B2
    .jpg','./images/xml/B2.xml');
INSERT INTO `images` VALUES (44,'/var/www/html/madonne_web/images/C1
    .jpg','./images/xml/C1.xml');
INSERT INTO `images` VALUES (45,'/var/www/html/madonne_web/images/C2
    .jpg','./images/xml/C2.xml');
INSERT INTO `images` VALUES (46,'/var/www/html/madonne_web/images/D1
    .jpg','./images/xml/D1.xml');
```



```
INSERT INTO `images` VALUES (47,'/var/www/html/madonne_web/images/D2
    .jpg','./images/xml/D2.xml');
INSERT INTO `images` VALUES (48,'/var/www/html/madonne_web/images/D3
    .jpg','./images/xml/D3.xml');
INSERT INTO `images` VALUES (49,'/var/www/html/madonne_web/images/E0
    .jpg','./images/xml/E0.xml');
INSERT INTO `images` VALUES (50,'/var/www/html/madonne_web/images/E1
    .jpg','./images/xml/E1.xml');
INSERT INTO `images` VALUES (51,'/var/www/html/madonne_web/images/E2
    .jpg','./images/xml/E2.xml');
INSERT INTO `images` VALUES (52,'/var/www/html/madonne_web/images/H1
    .jpg','./images/xml/H1.xml');
INSERT INTO `images` VALUES (53,'/var/www/html/madonne_web/images/H2
    .jpg','./images/xml/H2.xml');
INSERT INTO `images` VALUES (54,'/var/www/html/madonne_web/images/H3
    .jpg','./images/xml/H3.xml');
INSERT INTO `images` VALUES (55,'/var/www/html/madonne_web/images/I1
    .jpg','./images/xml/I1.xml');
INSERT INTO `images` VALUES (56,'/var/www/html/madonne_web/images/I2
    .jpg','./images/xml/I2.xml');
INSERT INTO `images` VALUES (57,'/var/www/html/madonne_web/images/G1
    .jpg','./images/xml/G1.xml');
INSERT INTO `images` VALUES (58,'/var/www/html/madonne_web/images/F1
    .jpg','./images/xml/F1.xml');
INSERT INTO `images` VALUES (59,'/var/www/html/madonne_web/images/G2
    .jpg','./images/xml/G2.xml');
INSERT INTO `images` VALUES (60,'/var/www/html/madonne_web/images/G3
    .jpg','./images/xml/G3.xml');
INSERT INTO `images` VALUES (61,'/var/www/html/madonne_web/images/I6
    .jpg','./images/xml/I6.xml');
INSERT INTO `images` VALUES (62,'/var/www/html/madonne_web/images/L1
    .jpg','./images/xml/L1.xml');
INSERT INTO `images` VALUES (63,'/var/www/html/madonne_web/images/L2
    .jpg','./images/xml/L2.xml');
INSERT INTO `images` VALUES (64,'/var/www/html/madonne_web/images/L4
    .jpg','./images/xml/L4.xml');
INSERT INTO `images` VALUES (65,'/var/www/html/madonne_web/images/N2
    .jpg','./images/xml/N2.xml');
INSERT INTO `images` VALUES (66,'/var/www/html/madonne_web/images/Z1
    .jpg','./images/xml/Z1.xml');
INSERT INTO `images` VALUES (67,'/var/www/html/madonne_web/images/O2
    .jpg','./images/xml/O2.xml');
INSERT INTO `images` VALUES (68,'/var/www/html/madonne_web/images/M2
    .jpg','./images/xml/M2.xml');
INSERT INTO `images` VALUES (69,'/var/www/html/madonne_web/images/V4
    .jpg','./images/xml/V4.xml');
INSERT INTO `images` VALUES (70,'/var/www/html/madonne_web/images/R2
    .jpg','./images/xml/R2.xml');
INSERT INTO `images` VALUES (71,'/var/www/html/madonne_web/images/S1
    .jpg','./images/xml/S1.xml');
INSERT INTO `images` VALUES (72,'/var/www/html/madonne_web/images/T1
    .jpg','./images/xml/T1.xml');
INSERT INTO `images` VALUES (73,'/var/www/html/madonne_web/images/Q3
    .jpg','./images/xml/Q3.xml');
```



```
INSERT INTO 'images' VALUES (74,'/var/www/html/madonne_web/images/Q2
    .jpg','./images/xml/Q2.xml');

CREATE TABLE 'signatures' (
  'index' tinyint(4) NOT NULL auto_increment,
  'name' varchar(30) NOT NULL default '',
  'label' varchar(35) NOT NULL default '',
  'weight' double NOT NULL default '1',
  'bin_path' varchar(255) NOT NULL default '',
  'type' varchar(30) default NULL,
  'nb_params' tinyint(4) default NULL,
  PRIMARY KEY  ('index'),
  UNIQUE KEY 'bin_path' ('bin_path'),
  KEY 'name' ('name')
) TYPE=MyISAM PACK_KEYS=0 AUTO_INCREMENT=9 ;

INSERT INTO 'signatures' VALUES (2,'MST','MST',1,'./cgi-bin/MST',
    NULL,NULL);
INSERT INTO 'signatures' VALUES (1,'Vinet','Vinet',1,'./cgi-bin/
    Vinet',NULL,NULL);
```

Voilà pour ce qui est de la base MySQL.

Faisons de même avec l'annuaire LDAP.

B.4.3 Configuration et installation de l'annuaire LDAP

Après avoir installé OpenLDAP, il faut configurer le serveur et créer l'annuaire. Voici la configuration du serveur LDAP située dans le fichier */etc/openldap/slapd.conf* :

```
include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/nis.schema

schemacheck on

allow bind_v2

pidfile /var/run/slapd.pid
argsfile /var/run/slapd.args

database bdb
suffix "dc=my-domain,dc=com"
rootdn "cn=Manager,dc=my-domain,dc=com"
rootpw {crypt}AHDcKj1mIqfPY

directory /var/lib/ldap

index objectClass                      eq,pres
index ou,cn,mail,surname,givenname     eq,pres,sub
index uidNumber,gidNumber,loginShell   eq,pres
index uid,memberUid                    eq,pres,sub
index nisMapName,nisMapEntry           eq,pres,sub
```



```
access to attr=userPassword
        by self write
        by anonymous auth
        by dn="cn=Manager,dc=my-domain,dc=com" write
        by * compare

access to *
        by dn="cn=Manager,dc=my-domain,dc=com" write
        by * read
```

Relancez le démon LDAP, comme ceci : # /sbin/service ldap restart. Ensuite, insérons les différents utilisateurs de la plate-forme. Pour cela nous allons utiliser un outil fort pratique a installer sur la machine, phpLDAPadmin.

Ouvrez dans un navigateur phpLDAPadmin avec les droits d'administration pour pouvoir créer les entrées de l'annuaire, puis sélectionnez l'annuaire et cliquez sur *Importer*. Sélectionnez le fichier LDIF associé à la plate-forme et cliquez sur *Continuer*. Voilà pour ce qui est de l'annuaire LDAP.

B.4.4 Paramétrage de la plate-forme

Comme nous l'avons vu dans le rapport, l'application est entièrement paramétrable par le biais de deux fichiers nommés *env_vars.php* et *config.php*.

Le fichier *config.php* concerne les identifiants, les mots de passe et les adresses IP des serveurs LDAP et MySQL, modifiez alors les différents champs pour qu'ils correspondent à la configuration mise en place.

Le fichier *env_vars.php* contient des variables relatives au mode de fonctionnement de la plate-forme, comprenant l'activation du mode XML-RPC, du mode de débogage (permettant d'afficher des informations supplémentaires pour résoudre d'éventuelles erreurs), etc... Modifiez donc les variables à votre convenance.

Annexe C

La Bibliothèque OpenCV d'Intel

C.1 Présentation

Open Source Computer Vision Library [1] est une bibliothèque de traitements d'images et de vision par ordinateur en langage C/C++, optimisée proposée par Intel pour Windows, Linux et comprend un très grand nombre d'opérateurs "classiques". Parmi lesquels :

- Création/libération d'images, macros d'accès rapide aux pixels
- Opérateurs standards :
  - morphologie
  - filtres dérivatifs, filtres de contours
  - suppression de fond
  - recherche de coins
  - Recherche, manipulation, traitement de contours
- Pyramides d'images
- Dessins de primitives géométriques (lignes, rectangles, ellipses, polygones... et même du texte)
- Création et utilisation d'histogrammes
- Changements d'espaces de couleurs (RGB, HSV, L*a*b* et YCrCb)
- Interface Utilisateur :
  - Lecture/écriture d'images (JPEG, PPM, ...)
  - Affichage à l'écran
  - Gestion des signaux sur clic de fenêtre, fermeture, ...
- Opérateurs plus complexes
- Mean Shift
- Contours actifs
- Prédiction : Kalman et CONDENSATION
- Analyse du mouvement : Flot optique, MHI
- Détection de visages
- Calibrage de caméras (possible à partir d'un échiquier)
- Suivi d'objets 3D avec plusieurs caméras
- Mise en correspondance entre deux images
- lecture d'images à la volée directement depuis une vidéo AVI ou une caméra

Elle permet en outre de manipuler les structures de données suivantes :

- Tableaux, matrices et matrices creuses + opérateurs associés :
  - Opérateurs "classiques" de matrices (inversion, déterminant, transposée, valeurs/vecteurs propres, distance de Mahalanobis, ...)

[1] http://www.intel.com/research/mrl/research/opencv/overview.htm



– Piles, files... génériques. Listes denses et listes creuses. Ensembles.
– Graphes
– Arbres
– Modèles de Markov Cachés (HMM) 1D et 2D, algorithme de Viterbi, ...

C.1.1 Utilisation de la bibliothèque

Avant toute manipulation, il faut pouvoir charger en mémoire une structure de données représentant l'image sur laquelle on veut pouvoir appliquer des traitements.

Création d'image

Déclaration et création se font donc de la manière suivante :

```
#include <cv.h>
IplImage *im;
im = cvCreateImage(cvSize(512,256), IPL_DEPTH_8U, 3);
/* Image 512 x 256 de unsigned char avec 3 canaux */
```

Remplissage : (codage par défaut = BGR)

L'initialisation ou l'accès aux pixels de l'image s'effectue de cette façon :

```
pteur = (unsigned char*)im->imageData;
fin = im->width*im->height;
for(i=0; i<fin; i++)
{
  *(pteur++)=255; /* Bleu */
  *(pteur++)=0;   /* Vert */
  *(pteur++)=0;   /* Rouge */
}
```

On remarque que l'accès aux pixels se fait en réalité sur chaque canal par le biais d'un pointeur sur un tableau d'*unsigned char*.

Libération de l'espace mémoire

Une fois l'ensemble des traitements effectués, il ne faut bien évidemment pas oublier de désallouer l'espace mémoire utilisé par l'image, ceci se faisant en utilisant la fonction suivante : cvReleaseImage(&im);

Lire et afficher une image

Outre la manipulation même des pixels de l'image, cette bibliothèque permet le chargement à partir de fichiers mais également une visualisation des images en mémoire.



– Lire une image :

```
#include <highgui.h>
IplImage *im;
im=cvLoadImage("mon_image.jpg", 1); /* (1) */
/* 1 => 3 canaux (0 => 1 seul, -1 => automatique) */
im=cvLoadImage("mon_image.jpg", 0); /* (2) */
```

– Afficher une image :

```
cvNamedWindow("Ma fenetre", CV_WINDOW_AUTOSIZE);
cvShowImage("Ma fenetre", im);
cvWaitKey(0); /* Attendre qu'une touche soit pressée */
```



FIG. C.1 – Ouverture et visualisation d'images par OpenCV

Dessins de primitives

La création de formes géométriques de base est également implémentée dans un ensemble de primitives. Il faut toutefois faire attention au repère de l'image :

```
cvRectangle(im,cvPoint(50,100),cvPoint(75,150),CV_RGB(255,0,0),3);
cvRectangle(im,cvPoint(25,35),cvPoint(50,50),CV_RGB(255,255,0),
    CV_FILLED);
```



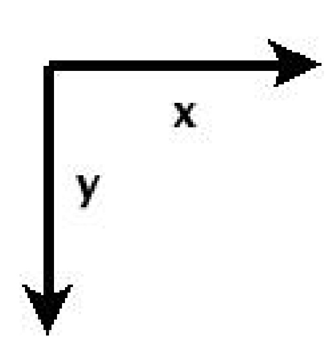


FIG. C.2 – Repère par défaut utilisé dans OpenCV

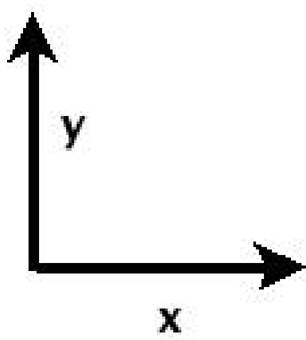


FIG. C.3 – *
Autre repère possible dans OpenCV

```
cvEllipse(im,cvPoint(105,100),cvSize(25,30),45.0,0,360,CV_RGB
    (100,160,220),3);
cvEllipse(im,cvPoint(115,10),cvSize(25,15),45.0,0,300,CV_RGB
    (220,150,20),CV_FILLED);
```





FIG. C.4 – Utilisation de primitives simples dans OpenCV

Il existe aussi de nombreuses autres fonctions de manipulations de primitives géométriques. Citons par exemple :

– renvoi de tous les points appartenant à un segment (délimité par ses deux extrémités)
– détection et traitement des contours

Détection de contours

Une des fonctionnalité particulièrement intéressante dans le contexte de la reconnaissance de formes et la recherche de descripteurs sur une image est bien entendu la détection de contours. Pour cela, un ensemble de filtres sont mis à la disposition du développeur :

– filtre de Canny,
– filtre de Sobel,
– filtre Laplacien,
– filtre de détection de coins,
– ...





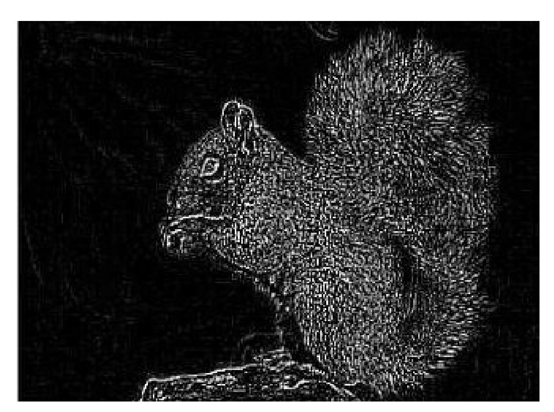


FIG. C.5 – Exemple obtenu à partir du filtre Laplacien dans OpenCV

```
im=cvLoadImage(argv[1],0);
im2=cvCreateImage(cvGetSize(im),IPL_DEPTH_16S,1);
cvLaplace(im,im2);
```

Différents type de transformées

Transformée en distances

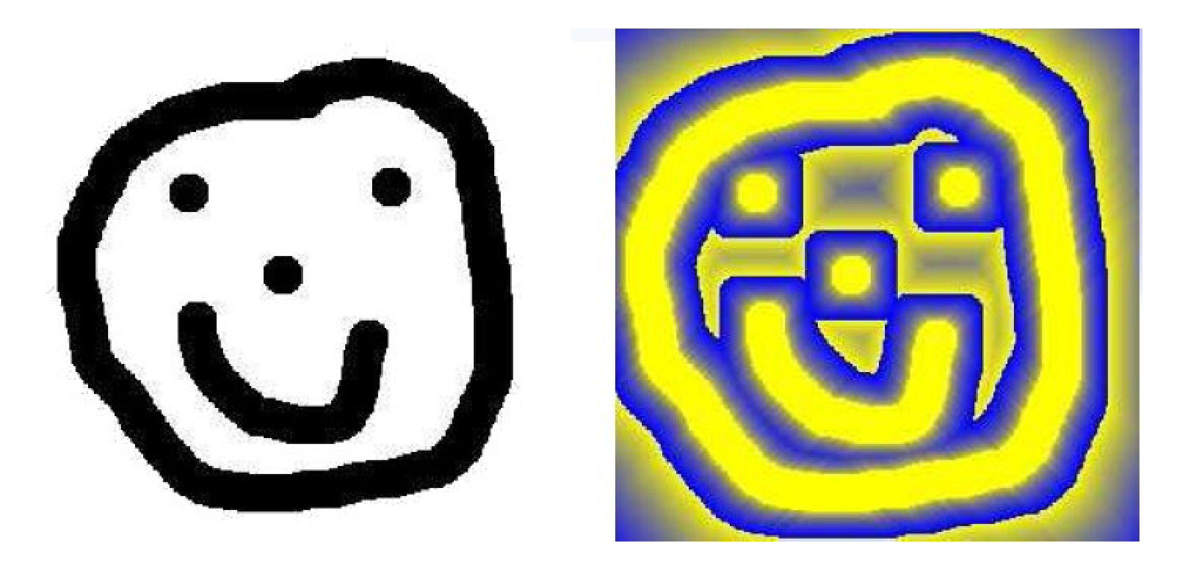


FIG. C.6 – Exemple utilisant la transformée de Borgefors dans OpenCV

Il s'agit de la transformée de Borgefors, avec plusieurs filtres :

```
CV_DIST_C (3x3): a=1, b=1
CV_DIST_L1 (3x3): a=1, b=2
CV_DIST_L2 (3x3): a=0.955, b=1.3693
CV_DIST_L2 (5x5): a=1, b=1.4, c=2.1969
```



On peut aussi définir ses propres filtres (en donnant a, b et c).

Transformée de Hough



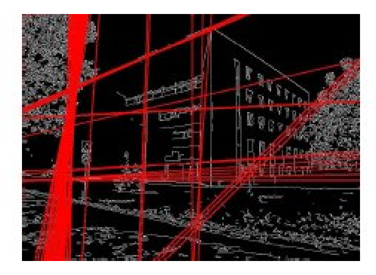




FIG. C.7 – Exemple de détection de lignes par la transformée de Hough dans OpenCV

Détection de lignes en utilisant le plan (ρ, Θ) :
- méthode "standard" (22° image) : renvoie un ensemble de droites
- méthode "probabiliste" (3e image) : renvoie un ensemble de segments

Transformée de Fourier et transformée cosinus

- transformée de Fourier (DFT) :
  - $\mathbb{R} \rightarrow \mathbb{R}$
  - $\mathbb{C} \rightarrow \mathbb{C}$
- transformée en cosinus (DCT) :
  - $\mathbb{R} \rightarrow \mathbb{R}$
- pour les deux :
  - transformées 1D et 2D
  - transformée inverse

```
imf=cvCreateImage(cvGetSize(im),IPL_DEPTH_32F,1);
dst=cvCreateImage(cvGetSize(im),IPL_DEPTH_32F,1);
cvConvert(im,imf);
cvDFT(imf,dst,CV_DXT_FORWARD);
```

Histogrammes

On peut également définir des histogrammes "standards" mais aussi des histogrammes "creux". La création n'est cependant pas directe : il faut auparavant désentrelacer les canaux de l'image.

```
/* Déclaration (on suppose que im existe et est définie */
IplImage* rgb[3];
int taillehist[3]={32,32,16};
```



```
/* Desentracelacement de l'image */
rgb[0]=cvCreateImage(cvGetSize(im),IPL_DEPTH_8U,1);
rgb[1]=cvCreateImage(cvGetSize(im),IPL_DEPTH_8U,1);
rgb[2]=cvCreateImage(cvGetSize(im),IPL_DEPTH_8U,1);
cvCvtPixToPlane(im,rgb[2],rgb[1],rgb[0],NULL);
/* Creation de l'histogramme */
hist=cvCreateHist(3,taillehist,CV_HIST_ARRAY,NULL,1);
cvCalcHist(rgb,hist,0,NULL);
/* Liberation */
cvReleaseHist(&hist);
```

Pour manipuler ces histogrammes, il existe des fonctions de comparaison d'histogrammes, traitement, rétroprojection sur une image, ...

Lecture d'une vidéo

Outre la manipulation d'images fixes, OpenCV est capable de lire une séquence d'images à partir d'un fichier vidéo au format AVI ou encore à partir d'un périphérique vidéo. Pour chaque étape, une image de type IplImage est automatiquement allouée (puis libérée) en mémoire.

```
/* Variables */
IplImage *im;
CvCapture *avi;
/* Ouverture de la video */
avi=cvCaptureFromAVI("ma_video.AVI");
while(cvGrabFrame(avi))
{
  im=cvRetrieveFrame(avi);
  /* Traitement de l'image */
}
```

Annexe D

Lightweight Directory Access Protocol

D.1 Présentation

Lightweight Directory Access Protocol (LDAP) est un protocole permettant l'accès des annuaires. LDAP est initialement un frontal d'accès à des bases d'annuaires respectant la norme X.500. Il est devenu un annuaire natif (standalone LDAP) utilisant sa propre base de données, sous l'impulsion d'une équipe de l'Université du Michigan.

LDAP est un protocole défini à l'IETF :
- Pour simplifier l'accès (consultation, modification) aux annuaires supportant les modèles d'information X.500 ;
- pour favoriser les implémentations et l'usage des annuaires.

Sa version courante est LDAP v3, définie dans la RFC 3377.

LDAP définit :
- Un protocole réseau pour accéder à l'information contenue dans l'annuaire ;
- un modèle d'information définissant la forme et le type de l'information contenue dans l'annuaire ;
- un espace de nommage définissant comment l'information est organisée et référencée ;
- un modèle fonctionnel définissant comment on accède et met à jour l'information ;
- un modèle de distribution permettant de répartir les données (à partir de la v3) ;
- un protocole et un modèle de données extensible ;
- des API pour développer des applications clientes.

Les données LDAP sont structurées dans une arborescence hiérarchique qu'on peut comparer au système de fichier Unix. Chaque noeud de l'arbre correspond à une entrée de l'annuaire ou *directory service entry* (DSE) et au sommet de cette arbre, appelé *Directory Information Tree* (DIT), se trouve la racine ou suffixe. Ce modèle est en fait repris de X500 mais, contrairement à ce dernier, conçu pour rendre un service d'annuaire mondial (ce que le DNS fait par exemple pour les noms de machines de l'Internet), l'espace de nommage d'un annuaire LDAP n'est pas inscrit dans un contexte global.



Les entrées correspondent à des objets abstraits ou issus du monde réel, comme une personne, une imprimante, ou des paramètres de configuration. Elles contiennent un certain nombre de champs appelés attributs dans lesquels sont stockées des valeurs. Chaque serveur possède une entrée spéciale, appelée *root directory specific entry* (rootDSE) qui contient la description de l'arbre et de son contenu.

Le schéma décrit les classes d'objets, les types des attributs et leur syntaxe. Chaque entrée de l'annuaire fait obligatoirement référence à une classe d'objet et ne doit contenir que des attributs qui sont rattachés au type d'objet en question. Une classe d'objet est définie par :

- Un nom qui l'identifie
- Un OID qui l'identifie également
- Des attributs obligatoires
- Des attributs optionnels
- Un type qui peut être structurel (objet courant de l'annuaire comme une personne), auxiliaire (permet d'ajouter des informations à des objets structurels) ou abstrait (objets basiques du schéma)

Annexe E

Webographie

E.1 Environnement d'accueil

Université de la Rochelle
http://www.univ-lr.fr

L3i - Laboratoire Informatique, Image, Interaction
http://www-l3i.univ-lr.fr/L3I/L3I.html

E.2 Administration réseau et système sous Unix

Léa / Linux entre amis, site d'aide Linux francophone
http://www.lea-linux.org/

Comment ça marche ? - site de vulgarisation de l'informatique.
http://www.commentcamarche.net/

E.3 Développement Web

PHP : Hypertext Preprocessor
http://www.php.net/

Nexen.net : Portail francais PHP et MySQL
http://www.nexen.net/

PHPIndex : La passerelle francaise des Technologies PHP
http://www.phpindex.com/

XML-RPC for PHP
http://phpxmlrpc.sourceforge.net/

E.4 XML et base de données

XMLfr : l'espace XML francophone
http://www.xmlfr.org/



eXist - Open Source Native XML Database
http://exist.sourceforge.net/

PHP and Perl classes to query eXist XML :DB
http://query-exist.sourceforge.net/

Présentation du MPEG-7
http://vlan.org/breve63.html

E.5 Traitement d'images

OpenCV : Open Source Computer Vision Library
http://www.intel.com/research/mrl/research/opencv/

ImageJ : Image processing and analysis in Java
http://rsb.info.nih.gov/ij/

E.6 Sources d'informations généralistes

Forums de discution axés programmation et/ou administration :
http://www.developpez.net/forums/index.php

JDN - Le Journal du Net
http://www.journaldunet.com/

01net - L'Actualité informatique
http://www.01net.com/

Wikipédia, l'encyclopédie libre et gratuite
http://fr.wikipedia.org/wiki/Accueil

Glossaire du jargon informatique :
http://www.linux-france.org/prj/jargonf/general/bgfrm.html